Tensor-GMRES Method for Large Sparse Systems of Nonlinear Equations

Dan Feng and Thomas H. Pulliam

The Research Institute for Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 212, Columbia, MD 21044 (410)730-2656

Work reported herein was supported by NASA under contract NAS 2-13721 between NASA and the Universities Space Research Association (USRA).

TENSOR-GMRES METHOD FOR LARGE SPARSE SYSTEMS OF NONLINEAR EQUATIONS*

by

Dan Feng† *and Thomas H. Pulliam*‡

*Work reported herein was supported by NASA under contract NAS 2-13721 between NASA and the Universities Space Research Association (USRA).

†Research Institute for Advanced Computer Science (RIACS), Mail Stop T20G-5, NASA Ames Research Center, Moffett Field, CA 94035-1000, USA. (feng@riacs.edu)

‡Fluid Dynamics Division, Mail Stop T27B-1, NASA Ames Research Center, Moffett Field, CA 94035-1000, USA. (pulliam@nas.nasa.gov)

Abstract

This paper introduces a tensor-Krylov method, the tensor-GMRES method, for large sparse systems of nonlinear equations. This method is a coupling of tensor model formation and solution techniques for nonlinear equations with Krylov subspace projection techniques for unsymmetric systems of linear equations. Traditional tensor methods for nonlinear equations are based on a quadratic model of the nonlinear function, a standard linear model augmented by a simple second order term. These methods are shown to be significantly more efficient than standard methods both on nonsingular problems and on problems where the Jacobian matrix at the solution is singular. A major disadvantage of the traditional tensor methods is that the solution of the tensor model requires the factorization of the Jacobian matrix, which may not be suitable for problems where the Jacobian matrix is large and has a "bad" sparsity structure for an efficient factorization. We overcome this difficulty by forming and solving the tensor model using an extension of a Newton-GMRES scheme. Like traditional tensor methods, we show that the new tensor method has significant computational advantages over the analogous Newton counterpart. Consistent with Krylov subspace based methods, the new tensor method does not depend on the factorization of the Jacobian matrix. As a matter of fact, the Jacobian matrix is never needed explicitly.

1 Introduction

This paper introduces a tensor-Krylov method for solving the nonlinear equations problem

$$\text{given } F : \Re^N \to \Re^N, \text{ find } x_* \in \Re^N \text{ such that } F(x_*) = 0. \tag{1.1}$$

Standard methods (such as Newton's method) widely used in practice for solving (1.1) are iterative methods that base each iteration upon a linear model of F at the current point x_c,

$$M(x_c + d) = F(x_c) + J_c d, \tag{1.2}$$

where $d \in \Re^N$ and $J_c \in \Re^{N \times N}$ is either the current Jacobian matrix or an approximation to it. When the size of J_c is moderate or the sparsity structure of J_c is favorable, many effective ways of solving the linear model are based on the factorization of J_c (for example LU factorization). However, for many real world problems (often arisen in numerical solution of ODEs and PDEs), J_c is often large and has a sparsity structure that does not allow a sparse factorization. The density of a full factorization would be so great that the storage of such a factorization would be impossible even on the most powerful computer available today due to unavoidable massive fill-ins. An attractive alternative to solving the linear model is the use of Krylov methods, such as the GMRES method, which does not require the factorization of J_c. The distinct advantage of Krylov methods is their minimum storage requirement and potential matrix-free implementations. Newton-like iteration schemes for solving the nonlinear equations problem using Krylov subspace projection methods as an inner linear solver are considered by many authors including Brown and Saad [5, 4], Chan and Jackson [6], and Brown and Hindmarsh [3]. Their computational results show that

these methods can be quite effective for many classes of problems in the context of systems of partial differential equations or ordinary differential equations.

The distinguishing feature of the Newton equation based algorithm is that if $F'(x_c)$ is Lipschitz continuous in a neighborhood containing the root x_*, $F'(x_*)$ is nonsingular and (1.2) is solved exactly (or to certain accuracy, e.g. see [4] for more details), then the sequence of iterates produced converges locally and q-quadratically to x_*. This means eventual fast convergence in practice. However, Newton's (or Newton-like) method is not usually quickly locally convergent, if $F'(x_*)$ is singular. This situation is analyzed and acceleration techniques are suggested by many authors, including Reddien [23], Decker and Kelley [8],[9],[10], Decker, Keller and Kelley [7], Kelley and Suresh [17], Griewank and Osborne [15], and Griewank [14]. In summary, their papers show that when the Jacobian at the solution has a null space of dimension one, then from good starting points, Newton's method is locally Q-linearly convergent with constant converging to $\frac{1}{2}$. The acceleration techniques presented in these papers depend upon *a priori* knowledge that the problem is singular.

Tensor methods for nonlinear equations introduced by Schnabel and Frank [26] are intended to be efficient both for nonsingular problems and for problems with low rank deficiency. These methods augment the standard linear model by a low rank second order term, in a way that requires no additional function or Jacobian evaluations per iteration, and hardly more arithmetic per iteration or total storage, than Newton's method. The second order term supplies higher order information in recent step directions; when the Jacobian is (nearly) singular, this usually results in supplying information in directions where the Jacobian lacks information or correspondingly, where the second order terms have the greatest influence. Tensor methods are shown to be considerably more efficient and robust than standard methods on both singular and nonsingular systems of nonlinear equations, with a larger margin of advantage on singular problems. As a matter of fact, Feng, Frank and Schnabel [13] show that on an appreciable class of singular problems, tensor model based methods exhibit multi-step (2-step or 3-step) q-superline convergence, whereas Newton's method has only linear convergence.

The traditional tensor methods are based on the factorization of the Jacobian matrix at each iteration, which makes them unsuitable for large systems of nonlinear equations where the factorization of the Jacobian matrix is too expensive. The goal of this paper is to develop a tensor-like iterative scheme for solving systems of nonlinear equations using Krylov subspace projection techniques. We will refer this method as the tensor-GMRES method. This method is independent of Jacobian factorization and can have Jacobian-free implementations. In addition, this method is intended to inherit the advantage of traditional tensor methods over the standard Newton's method both on singular and nonsingular problems.

Tensor-Krylov methods were first considered by Bouaricha in his Ph.D. thesis [2]. The basic idea is to solve the tensor model by calling a Krylov method for linear equations twice in each tensor iteration. Although the second call of the Krylov method might be less expensive (due to possible good initial guess) close to the solution, the computational

cost of one iteration of tensor methods based on this idea is likely twice as expensive as one iteration of analogous Newton-Krylov methods when away from the solution. This difficulty could make these tensor methods not competitive with its Newton counterpart in many situations.

This paper gives the tensor-GMRES method that requires no more function and derivative evaluations, and hardly more storage or arithmetic per iteration, than the analogous Newton-GMRES method. This is achieved by asking the tensor term in the tensor model to have a more restricted form than that in the traditional tensor model. The restriction imposed will have minimal impact on the performance of the tensor method, which is particularly true for problems where the Jacobian matrix is rank deficient or ill-conditioned at the solution. We discuss the formation and solution of the tensor model, and present our computational results. Like many Newton-Krylov algorithms, we show that the tensor-GMRES method introduced here can have efficient matrix free implementations.

We should point out that the basic idea of this paper can be used as a guidance for devising related tensor-Krylov methods such as tensor-Arnoldi, tensor-QMR and tensor-BiCG (these are under consideration by the authors). Due to nontrivial technical differences between these Krylov subspace based linear system solvers, we do not attempt to give a unified treatment of these tensor-Krylov methods, instead, we only concentrate on the tensor-GMRES method in this paper.

We would like to introduce some notation that will be used later on in this paper. We denote the solution to the system by x_*, and a current iterate by x_c or x_k throughout this paper. Consistent with tradition, we denote $F'(x)$ by $J(x)$ and usually abbreviate $J(x_c)$, $J(x_*)$ as J_c , J_* respectively. Similarly, we often abbreviate $F(x_c)$, $F(x_*)$, $F''(x_c)$, and $F''(x_*)$ as F_c , F_* , F_c'' , and F_*'' respectively. The notation $\|\cdot\|$ denotes the Euclidean vector norm. We use N to denote the length of x, which is the number of variables (equations also) in the system.

This paper is organized as follows. Section 2 briefly reviews the Arnoldi process, the GMRES algorithm and a line search Newton-GMRES algorithm. The traditional tensor methods for nonlinear equations are reviewed in Section 3. The main contribution of this paper is Section 4 which introduces the formation and solution of the new tensor-GMRES model. The implementation of the tensor-GMRES algorithm is given in Section 5. Comparative test results for our implementation of the tensor-GMRES method versus the analogous implementation of the Newton-GMRES method are also reported in this section. Finally, in Section 6, we summarize our research and make some brief comments on areas for future related research.

2 Newton-GMRES method for systems of nonlinear equations

The GMRES (Generalized Minimal RESidual) method was introduced by Saad and Schultz [24] for solving large unsymmetric systems of linear equations. This method is very effective when coupled with preconditioning techniques. It is also very competitive compared to other

iterative methods. Since the GMRES method is a Krylov subspace method, we first give a brief review of the Krylov subspace.

Given a matrix $A \in \Re^{N\times N}$, a vector $v_1 \in \Re^N$ and an integer $m \geq 1$, the Krylov subspace associated with A, v_1 and m is defined as

$$K_m(A, v_1) = span\{v_1, Av_1, A^2v_1, \cdots, A^{m-1}v_1\}.$$

Consider a system of linear equations

$$Ax = b. \tag{2.1}$$

Given an initial guess x_0 to the solution of the linear system, the initial residual is defined as

$$r_0 = Ax_0 - b.$$

The GMRES method attempts to find $z_m \in K_m(A, r_0)$ such that the residual vector $A(x_0 + z_m) - b$ is small, or in other words, $x_0 + z_m$ (approximately) solves (2.1). This is done in a fashion that at each iteration the residual norm is minimized.

The GMRES method is based on the Arnoldi process [1] which uses the Gram-Schmidt method to compute an l_2-orthonormal basis $\{v_1, v_2, \cdots, v_m\}$ of the $K_m(A, v)$ as follows.

Algorithm A: Arnoldi.

(A-1) Start. Choose a vector v_1 such that $\|v_1\| = 1$

(A-2) Iterate. For $j = 1, 2, \cdots$, do

$$\begin{aligned} h_{i,j} &= (Av_j, v_i), i = 1, 2, \cdots, j, \\ \hat{v}_{j+1} &= Av_j - \sum_{i=1}^{j} h_{i,j}v_i, \\ h_{j+1,j} &= \|\hat{v}_{j+1}\|, \\ v_{j+1} &= \hat{v}_{j+1}/h_{j+1,j}. \end{aligned}$$

As consequences of m iterations of the Arnoldi algorithm (assume it does not break down, i.e., $\|\hat{v}_{j+1}\|$ does not vanish throughout), we have $m + 1$ orthonormal vectors $v_1, \cdots, v_{m+1}$, and an $(m+1) \times m$ Hessenberg matrix $\bar{H}_m$ whose nonzero entries are given by $h_{i,j}$ produced by the algorithm. Let $V_m = [v_1, \cdots, v_m]$. As an important fact, the relation

$$AV_m = V_{m+1}\bar{H}_m$$

holds after each Arnoldi iteration.

The GMRES scheme is based on solving the least squares problem:

$$\min_{z_m \in K_m} \|f - A(x_0 + z_m)\| = \min_{z_m \in K_m} \|r_0 - Az_m\|, \tag{2.2}$$

where $r_0 = f - Ax_0$. If we set $z_m = V_m y$, $v_1 = r_0/\|r_0\|$ and $\beta = \|r_0\|$, this is equivalent to solving

$$\begin{aligned} & \min_{y \in \Re^m} \|\beta v_1 - AV_m y\| \\ = & \min_{y \in \Re^m} \|V_{m+1}(\beta e_1 - \bar{H}_m y)\| \\ = & \min_{y \in \Re^m} \|\beta e_1 - \bar{H}_m y\|. \end{aligned} \tag{2.3}$$

The least squares problem (2.3) is solved via a QR factorization of $\bar{H}_m$, which is fairly inexpensive because of the Hessenberg form of $\bar{H}_m$. When m is small the cost of solving (2.3) is minimal. Based on above observations, the GMRES algorithm for solving systems of linear equations is devised.

Algorithm G: GMRES.

(G-1) Start. Choose x_0 and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.

(G-2) Iterate. For $j = 1, 2, \cdots, m, \cdots$ until satisfied do:

$$\begin{aligned} h_{i,j} &= (Av_j, v_i), i = 1, 2, \cdots, j, \\ \hat{v}_{j+1} &= Av_j - \sum_{i=1}^{j} h_{i,j} v_i, \\ h_{j+1,j} &= \|\hat{v}_{j+1}\|, \\ v_{j+1} &= \hat{v}_{j+1}/h_{j+1,j}. \end{aligned}$$

(G-3) Form the approximate solution:
$x_m = x_0 + V_m y_m$ where y_m minimizes $\|\beta e_1 - \bar{H}_m y_m\|$, $y \in \Re^m$.

Due to memory limitation, it is necessary to restrain the number of Arnoldi iterations taken in (G-2). This leads to restarted versions of GMRES. The idea is to use the GMRES iteratively by restarting the algorithm every m steps, where m is some fixed integer parameter. Using the GMRES as a linear system solver, one can obtain a Newton-GMRES algorithm for nonlinear equations. At each iteration of the nonlinear algorithm, a (or an approximate) solution to the linear system

$$J_c d = -F_c, \tag{2.4}$$

with J_c being the current Jacobian matrix and F_c being the current function value, must be obtained. The Newton-GMRES method is an inexact Newton method, in the sense that at each iteration, the Newton-like step is obtained by solving the Newton equation approximately instead of "exactly". The step obtained in this way is required to be a descent direction for the function $\frac{1}{2}\|F(x)\|^2$. As a matter of fact, when the Newton equation is solved accurately enough (see [4] for details), the step obtained always gives a descent

direction for $\frac{1}{2}\|F(x)\|^2$. Afterwards, a global convergence strategy such as backtracking line search is employed to determine the step length along this descent direction, which will force progress towards the solution.

Algorithm NG: An iteration of the Newton-GMRES.

Given x_k, $J_k \in \Re^{N\times N}$ and $F_k \in \Re^N$.

(NG-1) Choose $\epsilon_k \in [0, 1)$.

(NG-2) Do GMRES (restart if necessary) to find $d^n = d_0 + V_m y^n$ such that

$$F_k + J_k d^n = r_k, \text{ with } \|r_k\|/\|F_k\| \leq \epsilon_k,$$

where d_0 is the initial guess to the solution of the Newton equation, and the columns of V_m form an orthonormal basis for the Krylov space generated by the Arnoldi process.

(NG-3) Find $\lambda > 0$ using a backtracking line search global strategy and form the next iterate $x_{k+1} = x_k + \lambda d^n$.

The residual vector r_k is the amount by which d^n fails to satisfy the Newton equation $J_k d + F_k = 0$. The forcing sequence ϵ_k is used to control the level of accuracy. The global and local convergence of inexact Newton methods is analyzed by Brown and Saad [4]. Their theory implies that if the sequence $\epsilon_k \to 0$, then under conditions (such as the Jacobian matrix is nonsingular at the solution) the iterates generated by Algorithm NG converges to the solution superlinearly; the convergence is quadratic if $\epsilon_k = O(\|F_k\|)$. This means eventual fast convergence in practice for nonsingular problems.

An attractive feature of Newton-Krylov algorithms is that the explicit computation of the Jacobian matrix is never needed. This is owing to the fact that the only computation involving the Jacobian matrix is the product of the Jacobian matrix and a vector, which can be approximated by finite difference

$$J(x)v = \frac{F(x + \sigma v) - F(x)}{\sigma}, \tag{2.5}$$

with a small number σ (see e.g. [11] for details).

3 Traditional tensor methods for nonlinear equations

The tensor model introduced by Schnabel and Frank [26] is a quadratic model of $F(x)$ formed by adding a second order term to a linear model, giving

$$M_T(x_c + d) = F(x_c) + J_c d + \frac{1}{2} T_c dd, \tag{3.1}$$

where $T_c \in \Re^{N\times N\times N}$ is intended to supply second order information about $F(x)$ around x_c. The second derivative of $F(x)$ at x_c, $F''(x_c) \in \Re^{N\times N\times N}$ is an obvious choice for T_c in

(3.1). However this choice for T_c has several serious disadvantages that preclude its use in practice. These include the computation of $N^3/2$ second partial derivatives of $F(x)$ and a storage requirement of at least $N^3/2$ real numbers for $F''(x_c)$. Furthermore, to utilize the model (3.1) with $T_c = F''(x_c)$, at each iteration one would have to solve a system of N quadratic equations in N unknowns, which is expensive and might not have a root.

The difficulties associated with the use of $T_c = F''(x)$ in (3.1) are overcome in tensor methods by choosing T_c to have a restricted low rank form. This can be considered as an extension to second order objects of the low rank update methods used to approximate Jacobian or Hessian matrices in secant (quasi-Newton) methods. One difference is that for reasons of efficiency in arithmetic cost and storage, at each iteration the zero tensor is updated rather than the tensor from the previous iteration.

Formation of the tensor model in Schnabel and Frank [26] is based upon the interpolation of information from past iterates, and requires no additional function or derivative evaluations. This is done by selecting some set of independent past iterates $x_{-1}, \cdots, x_{-p}$ and requiring the model (3.1) to interpolate the function values $F(x_{-k})$ at these points. That is, the model is required to satisfy

$$F(x_{-k}) = F(x_c) + F'(x_c)s_k + \frac{1}{2}T_c s_k s_k, \quad k = 1, \cdots, p,$$

where $s_k = x_{-k} - x_c$, $k = 1, \cdots, p$. The directions $\{s_k\}$ are required to be strongly linearly independent, which usually results in p being 1 or 2, although an upper bound of $p \leq \sqrt{N}$ is permitted. Then T_c is chosen to satisfy

$$\begin{array}{l} \min_{T_c \in \Re^{N \times N \times N}} \|T_c\|_F \\ \text{subject to } T_c s_k s_k = z_k, k = 1, \cdots, p, \end{array} \tag{3.2}$$

where $\|T_c\|_F$, the Frobenius norm of T_c is defined by

$$\|T_c\|_F^2 \;=\; \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}(T_c[i,j,k])^2,$$

and $z_k \in \Re^N$ is defined as $z_k = 2(F(x_{-k}) - F(x_c) - F'(x_c)s_k)$. The solution of (3.2) is given by

$$T_c \;=\; \sum_{k=1}^{p} a_k s_k s_k, \tag{3.3}$$

where a_k is the kth column of $A \in \Re^{N \times p}$, A is defined by $A = ZM^{-1}$, $M \in \Re^{p \times p}$ is defined by $M[i,j] = (s_i^T s_j)^2$, $1 \leq i, j \leq p$, and $Z \in \Re^{N \times p}$ by column k of $Z = z_k$, $k = 1, \cdots, p$.

Substituting (3.3) into the tensor model (3.1) gives

$$M_T(x_c + d) \;=\; F(x_c) + F'(x_c)d + \tfrac{1}{2}\sum_{k=1}^{p} a_k (s_k^T d)^2. \tag{3.4}$$

The additional storage required by T_c is $2p$ N-vectors, for $\{a_k\}$ and $\{s_k\}$. In addition, the $2p$ n-vectors $\{x_{-k}\}$ and $\{F(x_{-k})\}$ must be stored. Thus the total extra storage required for the tensor model is at most $4N^{1.5}$ since $p \leq \sqrt{N}$, which is likely to be small compared to the N^2 storage required for the Jacobian. The entire process for forming T_c requires $N^2p + O(Np^2)$ multiplications and additions. The leading term comes from calculating the p matrix-vector products $F'(x_c)s_k$, $k = 1, \cdots, p$; the cost of solving $A = ZM^{-1}$ is $O(Np^2)$. Since $p \leq \sqrt{N}$, the leading term in the cost of forming the tensor model is at most $N^{2.5}$ multiplications and additions per iteration, but it is usually only a small multiple of N^2 arithmetic operations in the dense case since p is usually 1 or 2. This cost also is small compared to the at least $N^3/3$ multiplications and additions per iteration required for the dense matrix factorizations by standard methods that use analytic or finite difference derivatives. If $F'(x)$ is large and sparse, the additional cost is simply p matrix-vector products, which generally is small in comparison to the costs of a standard iteration.

Solution of the tensor model with the special form of T_c given by (3.3) also can be performed efficiently in terms of algorithmic operations. The goal is to find a root of the tensor model (3.4), that is,

$$\text{find } d \in \Re^N \text{ such that}$$

$$M_T(x_c + d) = F(x_c) + F'(x_c)d + \tfrac{1}{2}\sum_{k=1}^{p} a_k(s_k^T d)^2 = 0. \tag{3.5}$$

Since (3.5) may not have a root, it is generalized to solving

$$\min_{d \in \Re^N} \|M_T(x_c + d)\|_2. \tag{3.6}$$

Schnabel and Frank [26] show that the solution of (3.6) can be reduced, in $O(N^2p)$ operations, to the least squares solution of a system of q quadratic equations in p unknowns, plus the solution of a system of $N-q$ linear equations and $N-p$ unknowns. (Usually, $q = p$; the exceptional case $q > p$ arises when the system of $N - p$ linear equations and $N - p$ unknowns would be singular and generally only occurs when $rank(F'(x_c)) < N - p$.) This reduction is carried out by performing orthogonal transformations of both the variable and equation spaces in a way that isolates the quadratic terms into only p equations. The details of this process are not important to this paper, because here we deal solely with models where $p = 1$, in which case the tensor model can be solved much more simply and in closed form. For these reasons we do not discuss the solution algorithm further in this paper; for details, see [26]. The total cost of solving the tensor model is about $2N^3/3 + N^2p + O(N^2)$ multiplications and additions in the dense case, at most $N^2p \leq N^{2.5}$ multiplications more than the QR factorization of an $N \times N$ matrix. The process generally is numerically stable even if $F'(x_c)$ is singular but has rank $\geq N - p$. If $F'(x_c)$ is nonsingular, the Newton step can be obtained very cheaply as a by-product of the tensor model solution process. If $F'(x_c)$ is large and sparse, the tensor model solution still costs very little more than the standard Newton iteration, see [2].

In practice, computational results in [26] show the tensor method is more efficient than an analogous standard method based upon Newton's method on both nonsingular and

singular problems, with a particularly large advantage on singular problems. In tests on a standard set of nonsingular test problems, the tensor method is almost always more efficient than the standard method and is never significantly less efficient. The average improvement by the tensor method is $21-23\%$, in terms of iterations or function evaluations, on all test problems, and $36-39\%$ on the harder problems where one method requires at least ten iterations. The tensor method is considerably more efficient than the standard method on problems with $rank(F'(x_*)) = N-1$; the average improvement is $40-43\%$ on all problems and $57-61\%$ on the harder problems. The advantage of the tensor method over the standard method on problems with $rank(F'(x_*)) = N-2$ is not as great as for the $rank(F'(x_*)) = N-1$ case but is still considerable, an average of $27-37\%$ improvement on all problems and $57-65\%$ on the harder problems. More recent computational experiments in [2], including experiments on much larger problems, show similar advantages for tensor methods.

It is shown in [13] that under mild conditions that the sequence of iterates generated by the tensor method based upon an ideal tensor model converges locally and two-step Q-superlinearly to the solution with Q-order $\frac{3}{2}$, and that the sequence of iterates generated by the tensor method based upon a practical tensor model converges locally and three-step Q-superlinearly to the solution with Q-order $\frac{3}{2}$. In the same situation, it is known that standard methods converge linearly with constant converging to $\frac{1}{2}$. Hence, tensor methods have theoretical advantages over standard methods. The analysis in [13] also confirms that tensor methods converge at least quadratically on problems where the Jacobian matrix at the root is nonsingular.

4 Tensor-GMRES method for nonlinear equations

4.1 Introduction

As reviewed in the previous section, traditional tensor methods for nonlinear equations are more successful both in theory and in practice than standard methods, notably for their fast local convergence and their efficiency in arithmetic cost per iteration. Nevertheless, since these methods are based on the factorization of the Jacobian matrix (or its approximation), they may not be attractive for many classes of large and sparse systems. Newton-Krylov methods are very effective in many situations for nonlinear equations problems. They often have fast local convergence as analyzed by Brown and Saad [4]. However, the fast local convergence can be impaired when the Jacobian matrix is singular or ill-conditioned at the solution. This is not uncommon in practice and accounts for a substantial amounts of the failures of Newton-like methods. The tensor method introduced here is primarily intended to improve upon the Newton-Krylov methods in cases where the Jacobian matrix is singular or ill-conditioned at the solution. This is done in a fashion similar to the Newton-GMRES method that avoids the requirement of the factorization of the Jacobian matrix by the traditional tensor methods. As a matter of fact the Jacobian matrix is never explicitly needed. In addition, the new method will have similar efficiency in arithmetic cost per

iteration to Newton-GMRES method.

The tensor model considered here uses only one past iterate information. There are three reasons for this. First, from the computational experience with tensor methods for nonlinear equations, the tensor methods that use one past point are easier to implement and have more satisfactory computational performance in practice. Second, from theoretical point of view, tensor methods based on single past point are better understood. Third and more importantly, the tensor model based on using more past points may require significantly more storage than the Newton-GMRES method since the latter only requires $O(N)$ (N being the size of the problem) memory locations and the storage of each past iterate information requires $O(N)$ locations as well. The situation is much different for the traditional tensor model based methods where p, the number of past points used, can be as high as $\sqrt{N}$.

The major difference between the new tensor model and the traditional tensor model is that the new tensor model has a more restricted second order term. The analysis of tensor methods for nonlinear equations by Feng, Frank and Schnabel [13] indicates that tensor methods will not lose fast local convergence on singular problems if the tensor term is projected into proper subspaces. As a matter of fact, we can show that for problems where the Jacobian matrix has rank deficiency one at the solution, if the second order term in the tensor model is projected into the subspace spanned by the left singular vector corresponding the smallest singular value of the Jacobian matrix, the theoretical results given in [13] remain intact. This means that methods based on the tensor model with the projected second order term will have fast local convergence on singular problems. This is the theoretical foundation of our tensor-GMRES method. The idea of projected tensor methods was first implemented in [12] for constrained optimization, where the projection is taken place in the variable space. The difference here is that the projection occurs in the function space.

This section is organized as follows. We first analyze two ideal tensor models that are closely related to our new tensor-GMRES model in Subsection 4.2. The formation and solution of the tensor-GMRES model is discussed in Subsections 4.3 and 4.4. We give a work comparison between the tensor-GMRES method and the analogous Newton-GMRES method in Subsection 4.7.

Some notation is also useful to us. Throughout this section we use N to denote the number of unknowns in the Newton equation and NZ to denote the number of nonzero elements in the Jacobian matrix.

Let $F'(x_c) = U_c D_c V_c^T$ be the singular value decomposition of $F'(x)$ at x_c, where $U_c = [u_1^c, u_2^c, \cdots, u_N^c]$, $V_c = [v_1^c, v_2^c, \cdots, v_N^c]$, and $D_c = diag[\sigma_1^c, \sigma_2^c, \cdots, \sigma_N^c]$, with $\sigma_1^c \geq \sigma_2^c \geq \cdots \geq \sigma_N^c \geq 0$ being the singular values of $F'(x_c)$ and $\{u_i^c\}$, $\{v_i^c\}$ being the corresponding left and right singular vectors.

Similarly, let $F'(x_*) = UDV^T$. Let v and u be the right and left singular vector of $F'(x_*)$ corresponding to the zero singular value, when $F'(x_*)$ has rank deficiency one.

4.2 Analysis of tensor methods using projected second order derivatives

The sequence of iterates produced by our algorithms is invariant to translations in the variable space. Thus no generality is lost by making the assumption that the solution occurs at $x_* = 0$, and this assumption is made throughtout this subsection. We also assume that v_N^c and u_N^c are so chosen that $\|v_N^c - v\| = O(\|x_c\|)$ and $\|u_N^c - u\| = O(\|x_c\|)$, whenever x_c is sufficiently close to x_*. This assumption is valid from the theorems about continuity of eigenvectors in Ortega [20] and Stewart [27], as long as $F'(x)$ is continuous near x_*, and has rank deficiency no greater than one at x_*.

Before going into the details of the analysis, we give Assumption 4.0, a group of assumptions that will be invoked for the remainder of this subsection in every result involving $F(x)$. These assumptions basically state that near x_*, the second order term supplies useful information in the null space direction of $F'(x_*)$, where $F'(x_*)$ lacks information.

Assumption 4.0 *Let $F : \Re^N \to \Re^N$ have two Lipschitz continuous derivatives. Let $F(x_*) = 0$, $F'(x_*)$ be singular with only one zero singular value, and let u and v be the left and right singular vector of $F'(x_*)$ corresponding to the zero singular value. Then we assume*

$$u^T F''(x_*)vv \quad \neq \quad 0 \tag{4.1}$$

where $F''(x_*) \in \Re^{N \times N \times N}$.

Assumption 4.0 is satisfied by most problems with $rank(F'(x_*)) = N - 1$, and has been assumed in most papers that analyze the behavior of Newton's method on singular problems. When $N = 1$, Assumption 3.0 is equivalent to $f''(x_*) \neq 0$.

Suppose we know the right and left singular vectors v_N^c and u_N^c corresponding to the least singular value of $F'(x_c)$ where x_c is the current iterate and $\|v_N^c\| = \|u_N^c\| = 1$. Then an excellent tensor model around x_c, if one is to utilize just a rank-one second order term, is

$$M_{T_{u_N}}(x_c + d) = F(x_c) + F'(x_c)d + \frac{1}{2}(u_N^c {u_N^c}^T)a_c(v_N^{cT} d)^2, \tag{4.2}$$

where $a_c = F''(x_c)v_N^c v_N^c$, because it contains the correct second order information where the Jacobian contains the least information, and correspondingly, where the second order term has the greatest influence.

Based on (4.2), a simple tensor algorithm, Algorithm PT, is designed.

Algorithm PT: Projected Tensor Algorithm.

IF (4.2) has real roots THEN

$d \leftarrow d_R$ where d_R solves $M_{T_{u_N}}(x_c + d) = 0$

ELSE $d \leftarrow d_M$ where d_M minimizes $\|M_{T_{u_N}}(x_c + d)\|$ □

Since (4.2) is the basis for our new tensor-GMRES model, we give an analysis of Algorithm PT.

Corollary 4.1 *Let Assumption 4.0 hold and $\{x_k\}$ be the sequence of iterates produced by Algorithm PT. There exist constants K_1, K_2 such that if $\|x_0\| \leq K_1$, then the sequence $\{x_k\}$ converges to x_* and*

$$\|x_{k+2}\| \leq K_2\|x_k\|^{\frac{3}{2}}$$

for $k = 0, 1, 2, \cdots$.

Proof. Note that $I = \sum_{i=1}^{N} u_i^c u_i^{cT}$ and $J_c = \sum_{i=1}^{N} \sigma_i^c u_i^c v_i^{cT}$. From the orthogonality of u_i^c, $i = 1, \cdots, N$, we have

$$\begin{aligned} & M_{T_{u_N}}(x_c + d) \\ = \; & (\sum_{i=1}^{N} u_i^c u_i^{cT})(F_c + \sum_{i=1}^{N} \sigma_i^c u_i^c v_i^{cT} d + \tfrac{1}{2}(u_N^c u_N^{cT}) a_c (v_N^{cT} d)^2) \\ = \; & [\sum_{i=1}^{N-1} (\sigma_i^c v_i^{cT} d + u_i^{cT} F_c) u_i^c] \\ & + (\sigma_N^c v_N^{cT} d + u_N^{cT} F_c + \tfrac{1}{2} u_N^{cT} a_c (v_N^{cT} d)^2) u_N^c. \end{aligned} \tag{4.3}$$

Note that the difference between (4.3) and (4.2) of [13] is only a second order term in the coefficient of each u_i^c for $i = 1, \cdots, N-1$, which does not effect either of the proofs of Lemmas 4.1 and 4.2 of [13]. The rest of the proof can be completed by following exactly the proof of Theorem 4.4 of [13]. □

Now we look at an interesting tensor model that is closely related to (4.2). Let $W \in \Re^{N \times m}$ with $m \leq N$ orthonormal columns. Consider the tensor model

$$M_{T_W}(x_c + d) = F(x_c) + F'(x_c)d + \frac{1}{2}(WW^T) a_c (v_N^{cT} d)^2, \tag{4.4}$$

where $u_N^c = Wy$ for $y \neq 0$, or u_N^c is in the span of the column vectors of W.

Corollary 4.2 *Let Assumption 4.0 hold and $\{x_k\}$ be the sequence of iterates produced by Algorithm PT with u_N^c being replaced by W. There exist constants K_1, K_2 such that if $\|x_0\| \leq K_1$, then the sequence $\{x_k\}$ converges to x_* and*

$$\|x_{k+2}\| \leq K_2\|x_k\|^{\frac{3}{2}}$$

for $k = 0, 1, 2, \cdots$.

Proof. Since $u_N^c = Wy$, from the orthogonality of columns of W, we have

$$u_N^c u_N^{cT} WW^T = u_N^c y^T W^T WW^T = u_N^c y^T W^T = u_N^c u_N^{cT}. \tag{4.5}$$

By similar reasoning as stated in the proof of Corollary 4.1, and using (4.5)

$$\begin{aligned} & M_{T_W}(x_c + d) \\ = \; & (\sum_{i=1}^{N} u_i^c u_i^{cT})(F_c + \sum_{i=1}^{N} \sigma_i^c u_i^c v_i^{cT} d + \tfrac{1}{2}(WW^T) a_c (v_N^{cT} d)^2) \\ = \; & [\sum_{i=1}^{N-1} (\sigma_i^c v_i^{cT} d + u_i^{cT} F_c + \tfrac{1}{2} u_i^{cT} (WW^T) a_c (v_N^{cT} d)^2) u_i^c] \\ & + (\sigma_N^c v_N^{cT} d + u_N^{cT} F_c + \tfrac{1}{2} u_N^{cT} a_c (v_N^{cT} d)^2) u_N^c. \end{aligned} \tag{4.6}$$

Again, note that the difference between (4.6) and (4.2) of [13] is only a second order term in the coefficient of each u_i^c for $i = 1, \cdots, N-1$, which does not effect either of the proofs of Lemmas 4.1 and 4.2 of [13]. The rest of the proof can be completed by following exactly the proof of Theorem 4.4 of [13]. □

4.3 Formation of the tensor model

At the current iterate x_c, assume that the Newton equation

$$F_c + J_c d^n = 0 \tag{4.7}$$

is solved by the GMRES method with $d^n = V_m y_m$ (assuming starting from zero). Let $\bar{H}_m$ be the $(m+1) \times m$ Hessenberg matrix from the Arnoldi process. An interesting fact is that the resulting Newton step d^n is in the span of the column vectors of V_m. The analysis of Feng, Frank and Schnabel [13] indicates that when the Jacobian matrix has a null space of dimension one at the solution, close to the solution, the Newton iterates fall into a funnel around the null space. In this situation, their theory also implies that the angle between d^n and v_n^c, the right singular vector corresponding to the smallest singular value of the current Jacobian matrix J_c, will be arbitrarily close to zero, close to the solution. As a consequence of $d^n = V_m y_m$, v_N^c will be arbitrarily close to being in the span of the column vectors of V_m. Consequently, u_N^c that is in the same direction as $J_c v_n^c$ will be arbitrarily close to being in the span of the column vectors of $J_c V_m$. Hence a good approximate to the projection matrix WW^T in (4.4) would be the projection matrix

$$P = (J_c V_m)[(J_c V_m)^T (J_c V_m)]^{-1} (J_c V_m)^T. \tag{4.8}$$

The singular vectors and the exact second order derivatives used in (4.4) are normally too expensive to obtain. We approximate them in the following manner. As in the situation of the traditional tensor model (3.4), let $s_c = x_p - x_c$, the difference between the past iterate x_p and the current iterate x_c. There are two choices for approximating v_N^c in (4.4), One is using $d^n/\|d^n\|$ since the Newton step d^n is likely to be along the null space close to the solution for singular problems. Another one is using $h = s_c/\|s_c\|$ since when consecutive iterates are in the funnel around the null space near the solution, the difference between the two consecutive iterates is also likely to be along the null space. We choose to use h because,

as we will see later, this will cause our tensor model to interpolate a past point in a projected function space. The ability of interpolating past points is vital for the success of traditional tensor methods for nonlinear equations on both singular and nonsingular problems. The term $a_c = F''(x_c)v_N^c v_N^c$ in (4.4) can be approximated by

$$a = \frac{2(F(x_p) - F(x_c) - J(x_c)s_c)}{s_c^T s_c} = F''(x_c)hh + E, \tag{4.9}$$

where $\|E\| = O(\|s_c\|)$. Equation (4.9) is standard in tensor model formation, which requires no extra function or Jacobian matrix evaluations.

Putting all the pieces together, we arrive at the following tensor model.

$$M_{T_P}(x_c + d) = F_c + J_c d + \tfrac{1}{2} P a (h^T d)^2, \tag{4.10}$$

where P is given by (4.8). It is easy to verify that the unprojected tensor model

$$M_T(x_c + d) = F_c + J_c d + \tfrac{1}{2} a (h^T d)^2 \tag{4.11}$$

interpolates the function value at the past point x_p. Hence, from

$$\begin{aligned} PM_{T_P}(x_c + d) &= PF_c + PJ_c d + \tfrac{1}{2} PPa(h^T d)^2 \\ &= PF_c + PJ_c d + \tfrac{1}{2} Pa(h^T d)^2 \\ &= P(F_c + J_c d + \tfrac{1}{2} a(h^T d)^2), \end{aligned}$$

the interpolation property of the full tensor model (4.11) implies that the projected tensor model (4.10) interpolates $F(x)$ at the past x_p in the subspace resulted from the projection of P onto the full function space. It is easy to see also that the tensor model (4.10) interpolates $F(x)$ at the current point in full function space. A second property of (4.10) is that when $m = N$, The projector matrix P is equal to identity, which recovers the full tensor model (4.11).

The formation of the tensor model (4.10) requires storage of two N-vectors F_p and x_p. The work of calculating a and h requires a matrix-vector multiplication costing NZ multiplications, and the evaluation of $\|s_c\|$ costing N multiplications. Since as we will see later, we do not have to form the projection matrix P explicitly, we will count the cost of calculations involving P to the cost of solving the tensor model, which will be addressed in the next subsection.

4.4 Solution of the tensor model

Solving the tensor model (4.10) in the full variable space is not preferable since it would be as expensive as solving the full tensor model. As discussed in Section 1, solving the full tensor model could involve calling a Krylov method for linear equations twice, which could be expensive in many situations. An alternative is to solve (4.10) in the Krylov subspace. When the Jacobian matrix lacks the first order information, close to the solution, the major

error of the current iterate would likely reside in the space spanned by v_N^c. Hence, we would like the tensor step to be along v_N^c so that it would have the biggest effect on reducing the error. Since v_N^c is arbitrarily close to being in the span of the columns of V_m, it is reasonable to require that the tensor step be in the span of the columns of V_m.

Therefore we would like to solve the least squares problem

$$\min_{z \in K_m} \|F_c + J_c z + \tfrac{1}{2} P a (h^T z)^2\|, \tag{4.12}$$

which is equivalent to solving

$$\min_{y \in \Re^m} \|F_c + J_c V_m y + \tfrac{1}{2} P a (h^T V_m y)^2\|. \tag{4.13}$$

Recall that $J_c V_m = V_{m+1} \bar{H}_m$. Let $\bar{H}_m = \bar{Q}_m \bar{R}_m$ be the QR factorization of $\bar{H}_m$. (Note that $\bar{Q}_m$ is the product of m Givens rotations; for details see [24].) From (4.8),

$$\begin{aligned} P &= V_{m+1} \bar{Q}_m \bar{R}_m [(V_{m+1} \bar{Q}_m \bar{R}_m)^T (V_{m+1} \bar{Q}_m \bar{R}_m)]^{-1} (V_{m+1} \bar{Q}_m \bar{R}_m)^T \\ &= V_{m+1} \bar{Q}_m \bar{R}_m (\bar{R}_m^T \bar{R}_m)^{-1} \bar{R}_m^T \bar{Q}_m^T V_{m+1}^T \\ &= V_{m+1} \bar{Q}_m \begin{pmatrix} I_m & 0 \\ 0 & 0 \end{pmatrix} \bar{Q}_m^T V_{m+1}^T. \end{aligned} \tag{4.14}$$

Using (4.14) and $r_0 = -F_c$, and letting b be the first m components of $\bar{Q}_m^T V_{m+1}^T a$, (4.13) is equivalent to

$$\begin{aligned} & \min_{y \in \Re^m} \| - r_0 + V_{m+1} \bar{H}_m y + \tfrac{1}{2} V_{m+1} \bar{Q}_m \begin{pmatrix} b \\ 0 \end{pmatrix} (h^T V_m y)^2 \| \\ = & \min_{y \in \Re^m} \|V_{m+1} (\|r_0\| e_1 - \bar{Q}_m \bar{R}_m y - \tfrac{1}{2} \bar{Q}_m \begin{pmatrix} b \\ 0 \end{pmatrix} (h^T V_m y)^2)\| \\ = & \min_{y \in \Re^m} \|\bar{Q}_m (\bar{Q}_m^T \|r_0\| e_1 - \bar{R}_m y - \tfrac{1}{2} \begin{pmatrix} b \\ 0 \end{pmatrix} (h^T V_m y)^2)\| \\ = & \min_{y \in \Re^m} \|w - \bar{R}_m^1 y - b (h^T V_m y)^2\| \end{aligned} \tag{4.15}$$

where w is the first m components of $\bar{Q}_m^T \|r_0\| e_1$ and $\bar{R}_m^1$ is the first m rows of $\bar{R}_m$. An interesting feature of (4.15) is that if the system of quadratics has a root, the minimum norm of the tensor model in the Krylov space will be the absolute value of the last component of $\bar{Q}_m^T \|r_0\| e_1$, which is the same as the residual norm of the Newton equation solved by the GMRES algorithm. When $\bar{R}_m^1$ is nonsingular, using the techniques for solving the tensor model developed in [2], we form the β function

$$\begin{aligned} q(\beta) &= h^T V_m (\bar{R}_m^1)^{-1} w - h^T V_m y - \tfrac{1}{2} h^T V_m (\bar{R}_m^1)^{-1} b (h^T V_m y)^2 \\ &= \hat{h}^T (\bar{R}_m^1)^{-1} w - \beta - \tfrac{1}{2} \hat{h}^T (\bar{R}_m^1)^{-1} b \beta^2, \end{aligned} \tag{4.16}$$

where $\beta = h^T V_m y$ and $\hat{h} = V_m^T h$, and solve the minimization problem

$$\min_{\beta \in \Re} \|q(\beta)\|. \tag{4.17}$$

Let β_* be a solution to (4.17). By the theory established in [2], (4.15) is solved by

$$y_*^t = [(\bar{R}_m^1)^T(\bar{R}_m^1)]^{-1}\hat{h}q(\beta_*)/\omega + (\bar{R}_m^1)^{-1}w - \tfrac{1}{2}(\bar{R}_m^1)^{-1}b\beta_*^2,$$

where $\omega = \hat{h}^T[(\bar{R}_m^1)^T(\bar{R}_m^1)]^{-1}\hat{h}$. Then the tensor step for the system of nonlinear equations is given by

$$d^t = V_m y_*^t. \tag{4.18}$$

Compared to the solution of the Newton equation by the GMRES algorithm, the solution of the tensor model requires a minimal amount of extra work. The major extra work comes from forming d^t from (4.18) and the calculation of $V_m^T h$ and $V_m^T a$, each requiring mN multiplications. In addition, forming the β function requires one extra backsolve of an $m \times m$ triangular system and two dot-products of vectors of length m, costing $\frac{1}{2}m^2 + 2m$ multiplications; forming b needs an applications of m Givens rotations, costing $4m$ multiplications; forming $[(\bar{R}_m^1)^T(\bar{R}_m^1)]^{-1}\hat{h}$ requires two backsolves of an $m \times m$ triangular system, costing m^2 multiplications, and then forming ω needs a dot-product of two m-vectors, costing m multiplications. In summary, the total extra cost of solving the tensor model is $3mN + \frac{3}{2}m^2 + 7m$.

4.5 Dealing with $d_0 \neq 0$

Because of the memory limitation, it is necessary for the GMRES algorithm to restart. As a result, the solution to the Newton equation obtained from the GMRES algorithm is given by $d^n = d_0 + V_m y^n$. In this situation, d^n is in the span of $\{d_0, V_m\}$, instead of in the span of $\{V_m\}$ only, as in the situation of $d_0 = 0$. For reasons discussed before, we would like the tensor step to be in the span of $\{d_0, V_m\}$. Consequently, we would like the projection matrix P to be onto the subspace spanned by $\{J_c V_m, J_c d_0\}$. One choice of such a P is

$$\bar{P} = M(M^T M)^{-1} M^T$$

where $M = J[V_m, d_0]$. Hence we would like to solve

$$\min_{d \in \{d_0\} \cup K_m} \|F_c + J_c d + \tfrac{1}{2}\bar{P}a(h^T d)^2\|. \tag{4.19}$$

Let $d = V_m y + d_0 \tau$. Using $r_0 = -F_c - J_c d_0$, (4.19) is equivalent to solving

$$\begin{aligned} &\min_{y \in \Re^m, \tau \in \Re} \|F_c + J_c[V_m, d_0]\begin{pmatrix} y \\ \tau \end{pmatrix} + \tfrac{1}{2}\bar{P}a\{h^T[V_m, d_0]\begin{pmatrix} y \\ \tau \end{pmatrix}\}^2\| \\ = &\min_{y \in \Re^m, \tau \in \Re} \|F_c + [J_c V_m, J_c d_0]\begin{pmatrix} y \\ \tau \end{pmatrix} + \tfrac{1}{2}\bar{P}a\{h^T[V_m, d_0]\begin{pmatrix} y \\ \tau \end{pmatrix}\}^2\| \end{aligned}$$

$$= \min_{y\in\Re^m,\tau\in\Re} \|F_c + [J_cV_m, -F_c - r_0]\begin{pmatrix} y \\ \tau \end{pmatrix} + \tfrac{1}{2}\bar{P}a\{h^T[V_m, d_0]\begin{pmatrix} y \\ \tau \end{pmatrix}\}^2\|$$

$$= \min_{y\in\Re^m,\tau\in\Re} \|F_c + [V_{m+1}\bar{H}_m, g]\begin{pmatrix} y \\ \tau \end{pmatrix} + \tfrac{1}{2}\bar{P}a\{h^T[V_m, d_0]\begin{pmatrix} y \\ \tau \end{pmatrix}\}^2\|, \qquad (4.20)$$

where $g = -F_c - r_0$. An important feature of (4.20) is that it does not involve the Jacobian matrix J_c. We simplify (4.20) to

$$\min_{\hat{y}\in\Re^{m+1}} \|F_c + \hat{J}\hat{y} + \tfrac{1}{2}\hat{a}(\hat{s}^T\hat{y})^2\|, \qquad (4.21)$$

where $\hat{J} = [V_{m+1}\bar{H}_m, g]$, $\hat{y} = \begin{pmatrix} y \\ \tau \end{pmatrix}$, $\hat{a} = \bar{P}a$ and $\hat{s} = [V_m, d_0]^T h$. We discuss how to form and solve (4.21) efficiently. The solution of this type of nonlinear least squares problem is studied by Schnabel and Bouaricha in [25]. Their theory shows that when $\hat{J}$ has full rank the solution of (4.21) is given by

$$\hat{y}_* = (\hat{J}^T\hat{J})^{-1}\hat{s}\; q(\beta_*)/\omega - (\hat{J}^T\hat{J})^{-1}\hat{J}^T(F_c + \tfrac{1}{2}\hat{a}\beta_*^2), \qquad (4.22)$$

where $q(\beta)$ and ω are defined by

$$q(\beta) = \hat{s}^T(\hat{J}^T\hat{J})^{-1}\hat{J}^T F_c + \beta + \tfrac{1}{2}\hat{s}^T(\hat{J}^T\hat{J})^{-1}\hat{J}^T\hat{a}\beta^2, \qquad (4.23)$$

$$\omega = \hat{s}(\hat{J}^T\hat{J})^{-1}\hat{s}, \qquad (4.24)$$

and the value of β_* is determined from

$$\min_{\beta\in\Re} \|q(\beta)/\sqrt{\omega}\|^2 + \|n(\beta)\|^2, \qquad (4.25)$$

where $\|n(\beta)\|^2 = \|(F_c - \bar{P}F_c) + \frac{1}{2}(\hat{a} - \bar{P}\hat{a})\beta^2\|^2$.

In our situation, since $\bar{P}$ is a projector matrix and

$$\hat{a} - \bar{P}\hat{a} = \bar{P}a - \bar{P}\bar{P}a = \bar{P}a - \bar{P}a = 0,$$

$n(\beta)$ is a constant function. Hence the minimization problem (4.25) is equivalent to

$$\min_{\beta\in\Re} \|q(\beta)\|. \qquad (4.26)$$

To obtain $\hat{y}_*$ the critical computational work comes from the factorization of $\hat{J}^T\hat{J}$, since when this factorization is available, all the computations involving $(\hat{J}^T\hat{J})^{-1}$ can be achieved through backsolves. For this reason, we discuss the factorization of $\hat{J}^T\hat{J}$. Recall that $\hat{J} = [V_{m+1}\bar{H}_m, g]$ and $\bar{H}_m = \bar{Q}_m\bar{R}_m$. Hence we have

$$\hat{J}^T\hat{J} = [V_{m+1}\bar{H}_m, g]^T[V_{m+1}\bar{H}_m, g]$$

$$
\begin{aligned}
&= \begin{pmatrix} \bar{H}_m^T \bar{H}_m & \bar{H}_m^T V_{m+1}^T g \\ g^T V_{m+1} \bar{H}_m & g^T g \end{pmatrix} \\
&= \begin{pmatrix} \bar{R}_m^T \bar{R}_m & \bar{H}_m^T V_{m+1}^T g \\ g^T V_{m+1} \bar{H}_m & g^T g \end{pmatrix} \\
&= \begin{pmatrix} \bar{R}_m^{1\,T} & 0 \\ w^T & \gamma \end{pmatrix} \begin{pmatrix} \bar{R}_m^1 & w \\ 0 & \gamma \end{pmatrix}
\end{aligned}
$$

where $\bar{R}_m^1$ is the first m rows of $\bar{R}_m$, $w = \bar{Q}_m^T V_{m+1}^T g$, and $\gamma = \sqrt{g^T g - w^T w}$. The factorization is possible since $\hat{J}^T \hat{J}$ is always at least positive semidefinite. After $\hat{y}_*$ is obtained, (4.19) is solved by

$$d^t = [V_m, d_0]\hat{y}_*.$$

However, the calculation of expressions involving $(\hat{J}^T \hat{J})^{-1}$ is impossible if $\gamma = 0$. We discuss how to overcome this difficulty. Since $\hat{J} = [V_{m+1}\bar{H}_m, g]$ and $V_{m+1}\bar{H}_m$ has full rank, g has to be in the span of $\{V_{m+1}\bar{H}_m\}$, which implies that $J_c d_0 = g$ is in the span of $J_c V_m = V_{m+1}\bar{H}_m$. When J_c has full rank, this implies that d_0 is in the span of $\{V_m\}$, which in turn implies that $d^n = d_0 + V_m y^n$ is in the span of $V_m y^n$. In this situation, based on previous discussions, we actually would like to solve the tensor model (4.10) in the Krylov subspace V_m only, i.e.

$$\min_{z \in K_m} \|F_c + J_c(d_0 + z) + \tfrac{1}{2} P a (h^T (d_0 + z))^2\|, \tag{4.27}$$

where P is given by (4.8), which is equivalent to solving

$$\min_{y \in \Re^m} \|F_c + J_c d_0 + J_c V_m y + \tfrac{1}{2} P a (h^T (d_0 + V_m y))^2\|. \tag{4.28}$$

Using $J_c V_m = V_{m+1}\bar{H}_m$, $\bar{H}_m = \bar{Q}_m \bar{R}_m$, $r_0 = -F_c - J_c d_0$ and (4.14), and letting b be the first m components of $\bar{Q}_m^T V_{m+1}^T a$, (4.28) is equivalent to

$$
\begin{aligned}
&\min_{y \in \Re^m} \| - r_0 + V_{m+1}\bar{H}_m y + \tfrac{1}{2} V_{m+1} \bar{Q}_m \begin{pmatrix} b \\ 0 \end{pmatrix} (h^T (d_0 + V_m y))^2 \| \\
= &\min_{y \in \Re^m} \| V_{m+1} (\|r_0\| e_1 - \bar{Q}_m \bar{R}_m y - \tfrac{1}{2} \bar{Q}_m \begin{pmatrix} b \\ 0 \end{pmatrix} (h^T (d_0 + V_m y))^2) \| \\
= &\min_{y \in \Re^m} \| \bar{Q}_m (\bar{Q}_m^T \|r_0\| e_1 - \bar{R}_m y - \tfrac{1}{2} \begin{pmatrix} b \\ 0 \end{pmatrix} (h^T (d_0 + V_m y))^2) \| \\
= &\min_{y \in \Re^m} \| w - \bar{R}_m^1 y - b (h^T (d_0 + V_m y))^2 \|
\end{aligned} \tag{4.29}
$$

where w is the first m components of $\bar{Q}_m^T \|r_0\| e_1$ and $\bar{R}_m^1$ is the first m rows of $\bar{R}_m$. Again using the techniques for solving the tensor model of nonlinear least squares developed in

[2], we form the β function

$$\begin{aligned} q(\beta) &= h^T V_m (\bar{R}_m^1)^{-1} w - h^T V_m y - \tfrac{1}{2} h^T V_m (\bar{R}_m^1)^{-1} b (h^T (d_0 + V_m y))^2 \\ &= \hat{h}^T (\bar{R}_m^1)^{-1} w + h^T d_0 - \beta - \tfrac{1}{2} \hat{h}^T (\bar{R}_m^1)^{-1} b \beta^2, \end{aligned} \tag{4.30}$$

where $\beta = h^T(d_0 + V_m y)$ and $\hat{h} = V_m^T h$, and solve the minimization problem

$$\min_{\beta \in \Re} \|q(\beta)\|. \tag{4.31}$$

Let β_* be a solution to (4.31). By the theory established in [2], (4.29) is solved by

$$y_*^t = [(\bar{R}_m^1)^T (\bar{R}_m^1)]^{-1} \hat{h} q(\beta_*)/\omega + (\bar{R}_m^1)^{-1} w - \tfrac{1}{2} (\bar{R}_m^1)^{-1} b \beta_*^2.$$

Then the tensor step for the system of nonlinear equations is given by

$$d^t = d_0 + V_m y_*^t. \tag{4.32}$$

We examine the cost of solving the tensor model when $d_0 \neq 0$. Since the situation of $\gamma = 0$ is less expensive to deal with, we will concentrate on the situation of $\gamma \neq 0$. Compared to the solution of the Newton equation by the GMRES algorithm in a similar situation, the solution of the tensor model requires a minimal amount of extra work. The major extra work comes from forming d^t, $\hat{s}$, $\hat{J}^T \hat{a}$ and $\hat{J}^T F_c$, each requiring $(m+1)N$ multiplications (note that $\hat{J}^T \hat{a} = \hat{J}^T P a = \hat{J}^T a$ from the definition of P). Since $V_{m+1}^T g = V_{m+1}^T(-F_c - r_0) = -V_{m+1}^T F_c - \|r_0\| e_1$, given $V_{m+1}^T F_c$, cost of $V_{m+1}^T g$ is only a single subtraction. Hence the cost of factorization of $\hat{J}^T \hat{J}$, which involves the calculation of $\bar{Q}_m^T V_{m+1}^T g$, $g^T g$ and $w^T w$, is $N + 5m$ multiplications. The operation count is accumulated from an application of m Givens rotations costing $4m$ multiplications, a dot-product of 2 N-vectors costing N multiplications and a dot-product of 2 m-vectors costing m multiplications. Given $\hat{s}$, $\hat{J}^T \hat{a}$ and $\hat{J}^T F_c$, the major cost of forming $q(\beta)$ and ω defined in (4.23) and (4.24) respectively, comes from the calculation of $\hat{s}^T (\hat{J}^T \hat{J})^{-1}$. Using the available factorization of $\hat{J}^T \hat{J}$, this can be done by two backsolves of $(m+1) \times (m+1)$ triangular systems, which costs $(m+1)^2$ multiplications. After $\hat{s}^T (\hat{J}^T \hat{J})^{-1}$ is obtained, the cost of forming $q(\beta)$ and ω is three dot-products of two m-vectors costing $3m$ multiplications totally. The cost of obtaining $\hat{y}_*$ using (4.22) needs two extra backsolves of $(m+1) \times (m+1)$ triangular systems, which costs $(m+1)^2$ multiplications, given $\hat{s}^T (\hat{J}^T \hat{J})^{-1}$, $\hat{J}^T \hat{a}$ and $\hat{J}^T F_c$. In summary, the total extra work required by solving the tensor model when the $d \neq 0$ is at most $(4(m+1)+1)N + 2(m+1)^2 + 8m$ multiplications.

4.6 Preconditioning and matrix free implementation

The success of the GMRES method on a system of linear equations usually depends on a good preconditioner. The formation and solution of the tensor model is consistent with preconditioning. When a preconditioner M is used in solving the Newton equation by the GMRES algorithm, it turned out that the only thing we need to do in the tensor step

calculation is to replace a by $M^{-1}a$, and F_c by $M^{-1}F_c$ (in case when $d_0 \neq 0$). The rest of the solution procedure is unchanged.

Compared to the Newton-GMRES method, the only extra computation involving the Jacobian matrix in the tensor-GMRES method is the computation of Js in the formation of the tensor term. In a Jacobian free implementation, this matrix-vector product can be approximated by the finite difference formula specified by (2.5). Hence the tensor-GMRES scheme is consistent with matrix free implementation.

4.7 Work comparison

If m steps of GMRES is required for solving the Newton equation, the computational cost of the Newton-GMRES iteration is $m(m+2)N + mNZ$ multiplications, and the storage requirement is $(m+2)N$ [24]. The extra storage required by the tensor-GMRES method is 2 N-vectors. As analyzed at the end of Subsection 4.3, the extra computational cost of forming the tensor model is $N + NZ$. As analyzed in Subsection 4.5, in the most expensive case the extra computational cost of solving the tensor model is $(4(m+1)+1)N + 2(m+1)^2 + 8m$ multiplications. Hence the combined extra computational cost of formation and solution of the tensor model in each tensor iteration, compared to an iteration of Newton-GMRES algorithm, is at most $(4(m+1)+2)N + 2(m+1)^2 + 8m + NZ$ multiplications. If we count the operations in flops (counting both multiplications and additions), the total extra computational operations are at about twice as many. Compared to $m(m+2)N + mNZ$, the cost of solving the Newton equation using the GMRES method, this extra cost is not significant if m is relatively large.

Because of the memory limitation, it is likely that m is not too large. Hence it is necessary to restart the GMRES algorithm. However, we should point out that the tensor model is not formed until the Newton equation is approximately solved by the GMRES algorithm, or in other words, until the Krylov space that contains the solution to the Newton equation is found. We form the tensor model only using the Krylov subspace generated in the last restarted GMRES algorithm. The tensor model has nothing to do with the intermediate Krylov spaces generated by the GMRES algorithm resulted from restarts before the final restart. Therefore compared to the total cost of the Newton-GMRES with restarts, the extra cost of formation and solution of the tensor model is likely to be minimal for a large portion of nonlinear problems, particularly hard problems that need many restarts of the GMRES algorithm.

5 Implementation and testing

In the previous section we presented the main new features of our tensor-GMRES method for nonlinear equations, namely, how we form the quadratic model of the nonlinear function, and how we solve this model efficiently. In this section first we give the complete algorithm we have implemented to test these ideas and clarify various aspects of this algorithm and its computer implementation. Then, we present some results of testing the method on several

problems.

This section is organized as follows. Subsection 5.1 gives a complete tensor-GMRES algorithm for systems of nonlinear equations and discusses the implementation of each step in details. In Subsections 5.2–5.4 we will show comparative test results for the tensor-GMRES algorithm, Algorithm TG, given in Subsection 5.1 versus the Newton-GMRES algorithm, Algorithm NG, given in Section 2 with the same implementation. Three distinct test problems., i.e., a Bradu problem, the Broyden Tridiagonal problem and the one-dimensional Euler equations problem, and several of their variants, were used in the testing. The tests on the Bradu problem, the Broyden Tridiagonal problem and their variants were performed on a Sun Super Workstation II+/50 at RIACS using MATLAB. The test on the one-dimensional Euler equations problem was performed on a Cray Y-MP at NAS facility using Fortran 90. Test results are summarized and discussed in Subsection 5.5.

5.1 A complete algorithm

This subsection presents Algorithm TG, the full algorithm of the tensor-GMRES method and discusses implementation issues of this algorithm in details.

Algorithm TG. An iteration of the tensor-GMRES method.

Given $x_k \in \Re^N$, x_{k-1}, $J_k \in \Re^{N\times N}$, $F_k \in \Re^N$ and $F_{k-1} \in \Re^N$.

(TG-1) Decide whether to stop. If not:

(TG-2) Set $s = x_{k-1} - x_k$, $a = 2(F_{k-1} - F_k - J_k s)/(s^T s)$ and $h = s/\|s\|$. Choose $\epsilon_k \in [0, 1)$.

(TG-3) Do GMRES (restart if necessary) to find $d^n = d_0 + V_m y^n$ such that

$$\|F_k + J_k d^n\| < \epsilon_k,$$

where d_0 is the starting point of the last restarted GMRES procedure, and the columns of V_m form an orthonormal basis for the Krylov space generated by the corresponding Arnoldi process. In addition, let $\bar{H}_m$ be the Hessenberg matrix generated from the Arnoldi process, and $\bar{H}_m = \bar{Q}_m \bar{R}_m$ be its QR-factorization. Let $\bar{R}_m^1$ be the first m rows of $\bar{R}_m$.

(TG-4) If $d_0 = 0$ or $d_0 \in \{V_m\}$ then

Solve

$$\min_{y\in\Re^m} \|F_k + J_k d_0 + J_k V_m y + \tfrac{1}{2}\bar{a}\{h^T(d_0 + V_m y)\}^2\|, \qquad (5.1)$$

where $\bar{a} = (J_k V_m)\{(J_k V_m)^T (J_k V_m)\}^{-1}(J_k V_m)^T a$, by first solving

$$\min_{\beta\in\Re} \|q_1(\beta)(= \hat{h}_1^T (\bar{R}_m^1)^{-1} w + h^T d_0 - \beta - \tfrac{1}{2}\hat{h}_1^T (\bar{R}_m^1)^{-1} b\beta^2)\|,$$

to obtain a solution β_*, where w is the first m components of $Q_m^T\|F_k + J_k d_0\|e_1$, b is the first m components of $Q_m^T V_{m+1}^T a$, and $\hat{h}_1 = V_m^T h$. Then the solution to (5.1) is given by

$$y_*^t = [(\bar{R}_m^1)^T(\bar{R}_m^1)]^{-1}\hat{h}_1 q_1(\beta_*)/\omega + (\bar{R}_m^1)^{-1}w - \tfrac{1}{2}(\bar{R}_m^1)^{-1}b\beta_*^2,$$

where $\omega = \hat{h}_1^T[(\bar{R}_m^1)^T(\bar{R}_m^1)]^{-1}\hat{h}_1$.

Form the tensor step $d^t = d_0 + V_m y_*^t$.

Otherwise ($d_0 \neq 0$ and $d_0 \notin \{V_m\}$),

Solve

$$\min_{\hat{y}\in\Re^{m+1}} \|F_c + \hat{J}\hat{y} + \tfrac{1}{2}\hat{a}(\hat{h}_2^T\hat{y})^2\|, \tag{5.2}$$

where $\hat{J} = [V_{m+1}\bar{H}_m, u]$,
$\hat{a} = (J[V_m, d_0])\{(J[V_m, d_0])^T(J[V_m, d_0])\}^{-1}(J[V_m, d_0])^T a$
and $\hat{h}_2 = [V_m, d_0]^T h$. This is done by first solving

$$\min_{\beta\in\Re} \|q_2(\beta)(= \hat{h}_2^T(\hat{J}^T\hat{J})^{-1}\hat{J}^T F_c + \beta + \tfrac{1}{2}\hat{h}_2^T(\hat{J}^T\hat{J})^{-1}\hat{J}^T\hat{a}\beta^2)\|$$

with solution β_*. Then the solution to (5.2) is given by

$$\hat{y}_* = (\hat{J}^T\hat{J})^{-1}\hat{h}_2 q_2(\beta_*)/\omega - (\hat{J}^T\hat{J})^{-1}\hat{J}^T(F_c + \tfrac{1}{2}\hat{a}\beta_*^2),$$

where $\omega = \hat{h}_2^(\hat{J}^T\hat{J})^{-1}\hat{h}_2$.

Form the tensor step $d^t = [V_m, d_0]\hat{y}_*$.

(TG-5) Choose a new step d from d^n and d^t.

(TG-6) Find $\lambda > 0$ using a backtracking line search global strategy and form the next iterate $x_{k+1} = x_k + \lambda d$.

Several tests are performed to determine whether to stop the algorithm in Step (TG-1). These stopping criteria are described by Dennis and Schnabel in Chapter 7 of [11]. For the sake of simplicity, we only use simplified versions of their criteria. The first test determines whether x_k solves the problem (1.1). This is accomplished by using $\|F(x_k)\| \leq$ FTOL, where a typical value of FTOL is around 10^{-5}, but the users can specify their own values for this tolerance. In our tests this value was set to 10^{-12} since we wanted to push the algorithms to their limits. The second test determines whether the algorithm has converged or stalled at x_k. It is done by measuring the relative change in the iterates from one step to the next. We use $\|x_k - x_{k-1}\|/\|x_{k-1}\| \leq$ STPTOL, where a typical STPTOL is around 10^{-8} in our implementation. Finally, we test if a maximum number of iterations is exceeded. Currently this value is 150.

In Step (TG-2), we need to choose the tolerance ϵ_k, which is passed to the GMRES algorithm when it is called to solve the Newton equation at the kth iteration. In [5], Brown and Saad suggested a sequence $\epsilon_k = \eta_k \|F(x_k)\|$, where $\eta_k = (\frac{1}{2})^k$ for $k = 1, 2, \cdots$. Since a good sequence is normally problem related, and again for the sake of simplicity, we use unchanging $\epsilon_k = 10^{-8}$ at every iteration for our test problems.

Step (TG-3) calls the normal GMRES linear equations solver. The number of Arnoldi iterations allowed between restarts is usually set to 20, and the number of maximum restarts of GMRES allowed is usually set to 150. However, these two values can be provided by the users based on their experience. When the tolerance is not reached after a maximum number of restarts, we simply go ahead and use the last computed data. When it returns, the GMRES algorithm readily provides $\bar{R}_m^1$ and $\bar{Q}_m$ that is presented in m Givens rotations. One by-product of Step (TG-3) is the Newton-GMRES step.

Step (TG-4) calculates the tensor-GMRES step. It is basically a concised reiterate of the solution of the tensor model described in the previous section. In Algorithm TG, the solutions of the two situations when $d_0 = 0$ and $d_0 \in \{V_m\}$, are combined for succinctness. The minimization of a quadratic function in one variable is done by using standard root formula. When a quadratic function has two distinct roots, the root that is smaller in absolute value is chosen.

Step (TG-5) usually consists of choosing the tensor step direction d^t obtained in Step (TG-4). However, the Newton step direction is chosen instead, when the tensor step direction is not a descent direction for $\frac{1}{2}\|F(x)\|^2$, which rarely occurs in practice but is not preluded in theory. Since the gradient of $\frac{1}{2}\|F(x)\|^2$ is $J(x)^T F(x)$, d^t is a descent direction if $(d^t)^T J(x)^T F(x) < 0$. We discuss how to compute this expression efficiently. At current iterate x_c, on the one hand, when $d_0 = 0$, using $d^t = V_m y^t$, $J_c V_m = V_{m+1} \bar{H}_m$ and $F_c = -\|F_c\| v_1$ yields

$$\begin{aligned} & (d^t)^T J_c^T F_c \\ = \; & (V_m y^t)^T J_c^T F_c \\ = \; & (y^t)^T (J_c V_m)^T F_c \\ = \; & (y^t)^T (V_{m+1} \bar{H}_m)^T F_c \\ = \; & (\bar{H}_m y^t)^T (V_{m+1}^T F_c) \\ = \; & -(\bar{H}_m y^t)^T \|F_c\| e_1. \end{aligned} \tag{5.3}$$

The cost of calculating (5.3) is minimal. On the other hand, when $d_0 \neq 0$, using $d^t = [V_m, d_0]\hat{y}^t$, $J_c V_m = V_{m+1} \bar{H}_m$ and $r_0 = -F_c - J_c d_0$ yields

$$\begin{aligned} & (d^t)^T J_c^T F_c \\ = \; & ([V_m, d_0]\hat{y}^t)^T J_c^T F_c \\ = \; & (\hat{y}^t)^T [J_c V_m, J_c d_0]^T F_c \\ = \; & (\hat{y}^t)^T [V_{m+1} \bar{H}_m, -r_0 - F_c]^T F_c \\ = \; & (\hat{y}^t)^T \begin{pmatrix} \bar{H}_m^T V_{m+1}^T F_c \\ -r_0^T F_c - \|F_c\|^2 \end{pmatrix} \end{aligned} \tag{5.4}$$

The major work in (5.4) is the calculation of $V_{m+1}^T F_c$. However, since this calculation is already done in the solution of the tensor step, no extra cost is necessary.

Finally, in Step (TG-6), we use a standard quadratic backtracking line search algorithm which is given below as Algorithm QB. The merit function $\frac{1}{2}\|F(x)\|^2$ is used for measuring the progress towards the solution.

Algorithm QB Standard quadratic backtracking line search algorithm.

Given a current point x_c, a search direction d, the directional derivative ξ, and $\alpha = 10^{-4}$.

(QB-1) Set $f_c = \frac{1}{2}\|F(x_c)\|^2$ and $x_+ = x_c + d$.

(QB-2) Set $f_+ = \frac{1}{2}\|F(x_+)\|^2$ and $\lambda = 1$.

(QB-3) While $f_+ > f_c + \alpha \cdot \lambda \cdot \xi$ do

$\lambda_{temp} = -\xi/(2[f_+ - f_c - \xi])$;

$\lambda = max\{\lambda_{temp}, \lambda/10\}$;

$x_+ = x_c + \lambda d$;

$f_+ = \frac{1}{2}\|F(x_+)\|^2$

EndWhile

When d^n is chosen in Step (TG-5), the directional derivative is given by $(d^n)^T J_c^T F_c$. Its calculation can be accomplished in a fashion similar to when d^t is chosen. When $d_0 = 0$, we can simply replace y^t in (5.3) by y^n and calculate $-(\bar{H}_m y^n)\|F_c\|e_1$. When $d_0 \neq 0$

$$\begin{aligned} & (d^n)^T J_c^T F_c \\ = \; & (d_0 + V_m y^n)^T J_c^T F_c \\ = \; & (J_c d_0 + J_c V_m y^n)^T F_c \\ = \; & (-F_c - r_0 + V_{m+1}\bar{H}_m y^n)^T F_c \\ = \; & -\|F_c\|^2 - r_0^T F_c + (y^n)^T \bar{H}_m^T (V_{m+1}^T F_c), \end{aligned}$$

which is easy to calculate given $V_{m+1}^T F_c$.

5.2 Test results for a Bradu problem and its variants

As a first test problem we choose to solve the nonlinear partial differential equation

$$-\Delta u = \lambda e^u \;\text{ in }\; \Omega, \quad u = 0 \;\text{ on }\; \Gamma, \tag{5.5}$$

where $\Delta = \nabla^2 = \sum_{i=1}^2 \partial^2/\partial x_i^2$ is the Laplace operator, $\Omega = (0,1) \times (0,1)$ and Γ is the boundary of Ω. This version of Bradu problem is chosen from a set of nonlinear model problems collected by Moré [18].

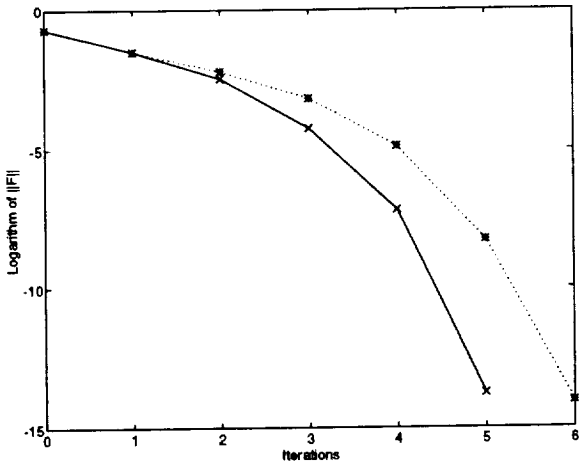

Figure 1: Results for the Bratu problem, $\lambda = 6.5$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

With n a positive integer we define $h = 1/(n+1)$ and then a mesh is given by

$$M_{ij} = \{ih, jh\}, \qquad 0 \leq i, j \leq n.$$

To approximate problem (5.5) we use the following finite-difference scheme:

$$\begin{aligned} -\frac{u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1} - 4u_{ij}}{h^2} &= \lambda e^{u_{ij}}, \qquad 1 \leq i, j \leq n \\ u_{kl} &= 0, \qquad \text{if} \quad M_{kl} \in \Gamma. \end{aligned} \tag{5.6}$$

In (5.6), u_{ij} is an approximation to $u(M_{ij})$. For $\lambda \leq 0$, (5.5) has a unique solution. For $\lambda > 0$, (5.5) may have one, several, or no solutions. In this test, we took $n = 32$ and $\lambda = 6.5$, which yields a system of $N = 1024$ equations in N unknowns. We tested the tensor-GMRES method given by Algorithm TG and the Newton-GMRES method given by Algorithm NG described in Section 2. In both algorithms, the standard diagonal preconditioning was used. The number of Arnoldi iterations allowed between restarts was set to 20 when the GMRES linear solver was called. The initial guess was chosen as $u_0 = 0$.

Figure 1 shows that the tensor-GMRES method has a slight improvement over the Newton-GMRES counterpart in number of nonlinear iterations. Although this problem is quite easy to solve for both methods and both methods exhibited quadratic convergence, the tensor-GMRES was converging a little bit faster. It took the tensor-GMRES method 5 iterations while it took the Newton-GMRES method 6 iterations to reach almost the same level of accuracy.

Since it is difficult to find large singular systems of nonlinear equations in the literature, we constructed all the singular test problems on our own. We give the test results for a rank one deficient modification of the Bradu problem. This singular problem is constructed by squaring the last equation in (5.6). The resulting problem has exactly the same solution

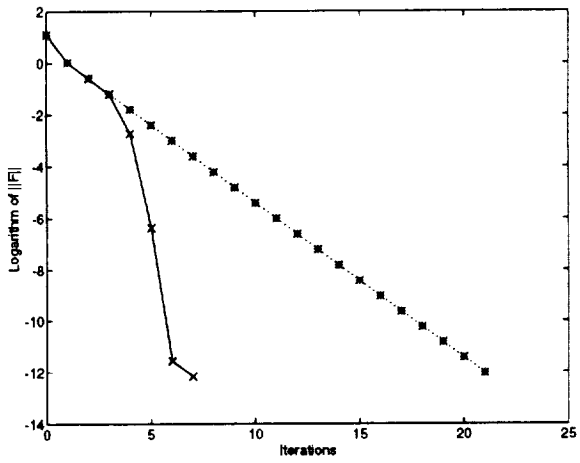

Figure 2: Results for the rank one deficient modification of the Bratu problem, $\lambda = -5$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

as the original problem. At the solution, the last row of the Jacobian matrix is a zero row. The rest of the rows are unchanged. Since the Jacobian matrix of the original problem is nonsingular, the new Jacobian matrix has rank deficiency one. In this test, we took $n = 32$ and $\lambda = -5$, which gives a system of $N = 1024$ equations in N unknowns. We tested the tensor-GMRES method given by Algorithm TG and the Newton-GMRES method given by Algorithm NG with the same implementation. In both algorithms, the standard diagonal preconditioning was used. Again when calling the GMRES routine we limited the number of Arnoldi iterations between restarts to 20. The initial guess was chosen as $u_0 = [1, 1, \cdots, 1]^T$.

Figure 2 shows that the tensor-GMRES method has a significant improvement over the Newton-GMRES counterpart in number of nonlinear iterations. The Newton-GMRES method is exhibiting linear convergence, while the tensor-GMRES method shows superlinear convergence. To reach the same accuracy, it took the tensor-GMRES method 7 iterations while it took the Newton-GMRES method 21 iterations. The margin of improvement in number of nonlinear iterations is more than 65%. One may notice the linear convergence behavior of the tensor-GMRES method at the last iteration. We suspect that this is caused by the round-off error.

Next we give the test results for a rank two deficient modification of the Bradu problem. Since the formation of the tensor model uses only one past point, one should expect the tensor method to help significantly with problems where the Jacobian matrix has rank deficiency one. It would be interesting to see whether the tensor method can help with problems where the Jacobian matrix has rank deficiency greater than one. This is the motivation for the rank two deficiency modification of the Bradu problem. This singular problem is constructed by squaring the last two equations in (5.6). Again the resulting problem has exactly the same solution as the original problem. At the solution, the last two rows of the Jacobian matrix are zero rows. The rest of the rows are unchanged. Since

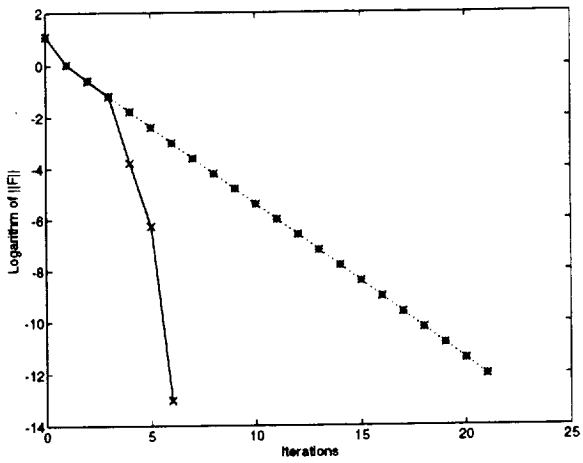

Figure 3: Results for the rank two deficient modification of the Bratu problem, $\lambda = -5$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

the Jacobian matrix of the original problem is nonsingular, the new Jacobian matrix has rank deficiency two. In this test, we took $n = 32$ and $\lambda = -5$, which yields a system of $N = 1024$ equations in N unknowns. We tested the tensor-GMRES method given by Algorithm TG and the Newton-GMRES method given by Algorithm NG under the same implementation. In both algorithms, the standard diagonal preconditioning was used. We limited the number of Arnoldi iterations between restarts to 20. The initial guess was chosen as $u_0 = [1, 1, \cdots, 1]^T$.

Figure 3 shows that the tensor-GMRES method again has a significant improvement over the Newton-GMRES counterpart in number of nonlinear iterations. The Newton-GMRES method is exhibiting linear convergence similar to the rank one deficient situation, while the tensor-GMRES method show superlinear convergence. The convergence pattern of the tensor-GMRES method is slightly different from the rank one deficient situation. The convergence here seems to be two-step superlinear, while in the rank one deficient situation the convergence seems to be one-step superlinear. It took the tensor-GMRES method 6 iterations while it took the Newton-GMRES method 21 iterations to reach almost the same level of accuracy. The margin of improvement is over 70% in number of nonlinear iterations.

5.3 Test results for the Broyden Tridiagonal problem and its variants

The Broyden Tridiagonal problem is chosen from a standard test set of Moré, Garbow and Hillstrom [19]. The function is defined as

$$f_i(x) = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1, \quad \text{for} \quad i = 1, \cdots, n \tag{5.7}$$

where $x_0 = x_{n+1} = 0$ and n can be any positive integer. A root of $f = 0$ is sought. For our test, we set $n = 1000$ which results in a system of 1000 nonlinear equations in 1000

unknowns. The Jacobian matrix has full rank at the solution. Three tests were performed on this problem. The standard starting point is $x_0 = [-1, -1, \cdots, -1]$. When calling the GMRES routine we limited the number of Arnoldi iterations between restarts to 20.

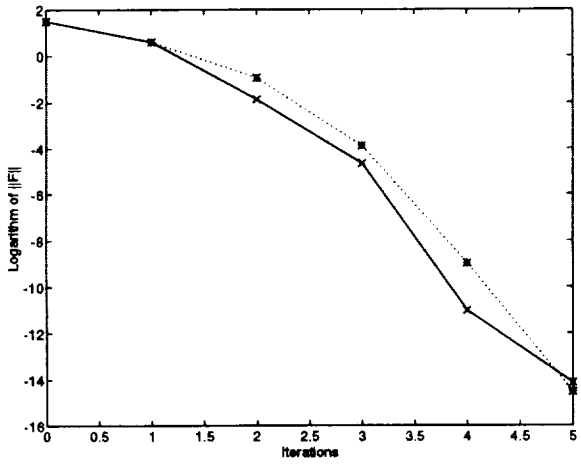

Figure 4: Results for the Broyden Tridiagonal problem. $x_0 = [-1, -1, \cdots, -1]^T$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

Figure 4 shows that the tensor-GMRES algorithm and the Newton-GMRES algorithm performed about the same, both exhibiting a quadratic convergence. Before the last step, the tensor-GMRES method was doing a little bit better, knocking down 1 or 2 more digits in function norm than the Newton-GMRES method. The last step of the tensor-GMRES broke the trend of the convergence. We believe that this is caused by the round off errors.

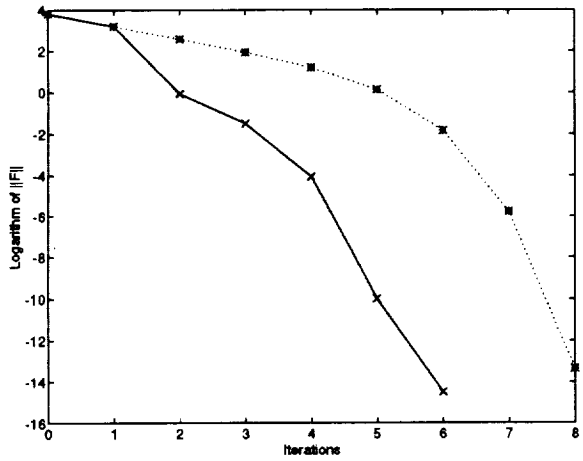

Figure 5: Results for the Broyden Tridiagonal problem. $x_0 = 10 * [-1, -1, \cdots, -1]^T$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

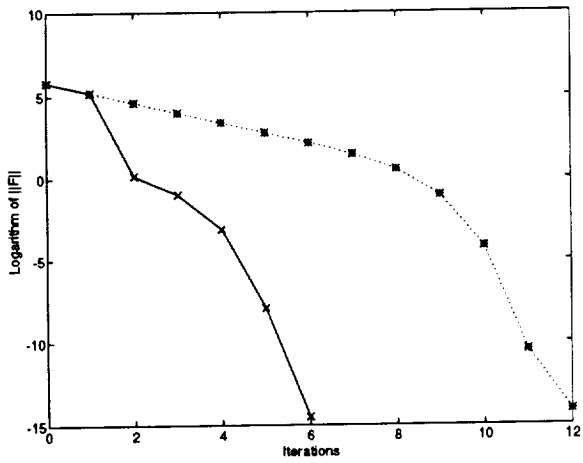

Figure 6: Results for the Broyden Tridiagonal problem. $x_0 = 100 * [-1, -1, \cdots, -1]^T$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

Since starting from the standard starting point x_0 seems too easy for both algorithms, we tried to start farther away from x_0.

Figure 5 shows the test results of starting from $10 * x_0$. We see that the tensor-GMRES method has a moderate improvement over the Newton-GMRES method in number of nonlinear iterations. To reach a similar accuracy, it took the tensor-GMRES method 6 iterations while taking the Newton-GMRES method 8 iterations.

Figure 6 shows the test results of starting from even farther with $100 * x_0$. This time we see that the tensor-GMRES method has a significant improvement over the tensor-GMRES method in number of nonlinear iterations. To reach a similar accuracy, it took the tensor-GMRES method 6 iterations while taking the Newton-GMRES method 12 iterations. The margin of improvement is 50%. It seems that for this problem the tensor-GMRES method is not sensitive to scaling up the starting point as the Newton-GMRES does.

Next we give the test results for a rank one deficient modification of the Broyden Tridiagonal problem. Again the problem was constructed by squaring the last function defined by (5.7). As discussed before this construction does not alter the solutions to the original system and results in a system whose Jacobian matrix has rank deficiency one at the solution. In this test, we took $n = 1000$, $x_0 = [-1, -1, \cdots, -1]$. When calling the GMRES routine we limited the number of Arnoldi iterations between restarts to 20.

Figure 7 shows that the tensor-GMRES method has a significant improvement over the Newton-GMRES method in number of nonlinear iterations. The Newton-GMRES method is exhibiting linear convergence, while the tensor-GMRES method shows superlinear convergence. To reach the same accuracy, it took the tensor-GMRES method 11 iterations while it took the Newton-GMRES method 22 iterations. The margin of improvement is 50%.

Finally, we give the test results for a rank two deficient modification of the Broyden

Tridiagonal problem. The problem was constructed by squaring the last two functions defined by (5.7). As discussed before this construction does not alter the solutions to the original system and results in a system whose Jacobian matrix has rank deficiency two at the solution. In this test, we took $n = 1000$, $x_0 = [-1, -1, \cdots, -1]$. When calling the GMRES routine we limited the number of Arnoldi iterations between restarts to 20.

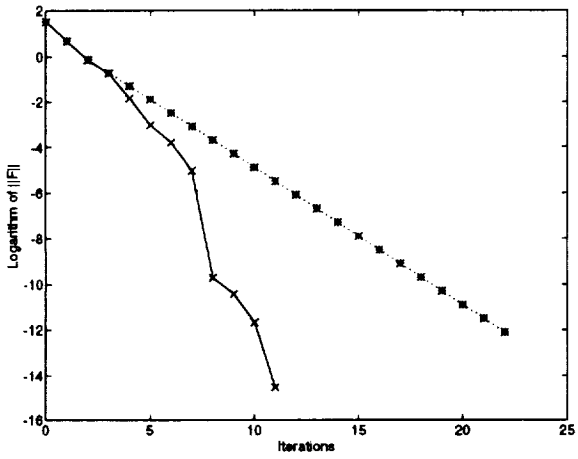

Figure 7: Results for a rank one deficient modification of the Broyden Tridiagonal problem. $x_0 = [-1, -1, \cdots, -1]^T$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

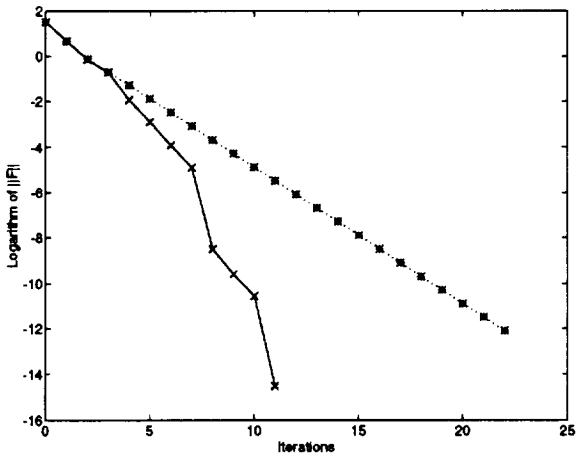

Figure 8: Results for a rank two deficient modification of the Broyden Tridiagonal problem. $x_0 = [-1, -1, \cdots, -1]^T$. Diagonal preconditioning. Solid line: tensor-GMRES; Dotted-line: Newton-GMRES.

Figure 8 shows that the tensor-GMRES method again has a significant improvement over the Newton-GMRES counterpart in number of nonlinear iterations. The Newton-GMRES

method is exhibiting linear convergence similar to the rank one deficient situation, while the tensor-GMRES method show superlinear convergence. The convergence pattern of the tensor-GMRES method again is slightly different from the rank one deficient situation. The convergence here seems to be two-step superlinear, while in the rank one deficient situation the convergence seems to be one-step superlinear. It took the tensor-GMRES method 11 iterations while it took the Newton-GMRES method 22 iterations to reach almost the same level of accuracy. The margin of improvement is 50% in number of nonlinear iterations.

5.4 Test results for one-dimensional Euler equations

One of the target applications for the tensor-Krylov methods is the nonlinear differential systems arising in physical problems, e.g. aerodynamics. One good model problem is the quasi-one-dimensional (1D) Euler equations for flow through a nozzle with a given area ratio. In particular, transonic conditions which generate a shock within the nozzle present a difficult test case, where methods typical of practical aerodynamic applications are required. Such methods include, finite difference, finite element, and unstructured grid finite volume techniques employing various forms of highly nonlinear algorithm constructions. For our purposes here, we have chosen one popular form of central finite differences with nonlinear artificial dissipation, see [21] for general details.

The quasi-1D Euler equations are

$$\mathbf{F}(\mathbf{Q}) = \partial_x \mathbf{E}(\mathbf{Q}) - \mathbf{H}(\mathbf{Q}) = 0 \quad 0.0 \leq x \leq 1.0 \tag{5.8}$$

where

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho v \\ e \end{bmatrix}, \quad \mathbf{E} = a(x) \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(e+p) \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 0 \\ -p\partial_x a(x) \\ 0 \end{bmatrix} \tag{5.9}$$

with ρ (density), u (velocity), e (energy), $p = (\gamma - 1)(e - 0.5\rho u^2)$ (pressure), $\gamma = 1.4$ (ratio of specific heats), and $a(x) = (1. - 4.(1 - a_t)x(1 - x))$ (the nozzle area ratio), with $a_t = 0.8$. For a given area ratio and shock location (here $x = 0.7$) an exact solution can be obtained from the method of characteristics.

We elect to use second order central differences

$$\partial_x u \approx \delta_x u_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x} \quad j = 0, \ldots, J_N \quad \Delta x = 1.0/J_N, \quad u_j = u(j\Delta x) \tag{5.10}$$

It is common practice and well known that artificial dissipation must be added to the discrete central difference approximations in the absence of any other dissipative mechanism, especially for transonic flows. Nonlinear dissipation as defined in [22], is used where 2^{nd} order, $D^2(\mathbf{Q})$, and 4^{th} order, $D^4(\mathbf{Q})$ difference formulas are employed.

$$D^2(\mathbf{Q}) = \nabla_x (\sigma_{j+1} + \sigma_j) \left(\epsilon_j^{(2)} \Delta_x \mathbf{Q}_j \right) \tag{5.11a}$$

$$D^4(\mathbf{Q}) = -\nabla_x (\sigma_{j+1} + \sigma_j) \left(\epsilon_j^{(4)} \Delta_x \nabla_x \Delta_x \mathbf{Q}_j \right) \tag{5.11b}$$

with

$$\nabla_\xi q_j = q_j - q_{j-1}, \quad \Delta_\xi q_j = q_{j+1} - q_j \tag{5.11c}$$

$$\epsilon_j^{(2)} = \kappa_2 \max(\Upsilon_{j+1}, \Upsilon_j, \Upsilon_{j-1})$$

$$\Upsilon_j = \frac{|p_{j+1} - 2p_j + p_{j-1}|}{|p_{j+1} + 2p_j + p_{j-1}|}$$

$$\epsilon_j^{(4)} = \max(0, \kappa_4 - \epsilon_j^{(2)}) \tag{5.11d}$$

where typical values of the constants are $\kappa_2 = 1/4$ and $\kappa_4 = 1/100$. The term $\sigma_j = |u| + c$ (where $c = \sqrt{(\gamma p/\rho)}$ is the speed of sound) is a spectral radius scaling.

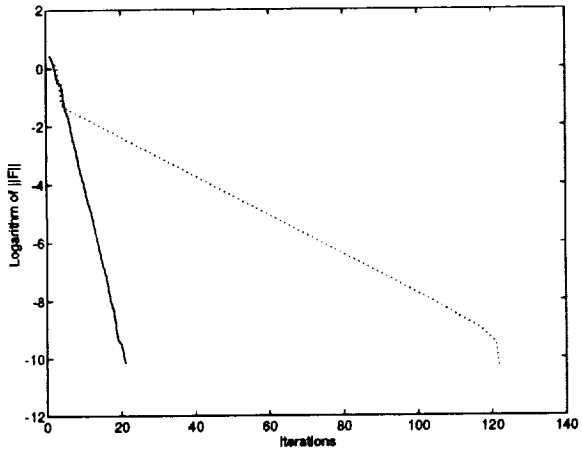

Figure 9: Results for 1D Euler using full nonlinear dissipation, Solid Line: tensor-GMRES; Dotted-line: Newton-GMRES.

Boundary operators at $j = 0$ and $j = J_N$ are defined in terms of physical conditions (taken from exact solution values) and the use of Riemann invariants. For this problem both inflow and outflow boundaries are subsonic and locally one-dimensional Riemann invariants are used. The locally one-dimensional Riemann invariants are given in terms of the velocity component as

$$R_1 = u - 2c/(\gamma - 1) \quad \text{and} \quad R_2 = u + 2c/(\gamma - 1). \tag{5.12}$$

The Riemann invariants R_1, R_2 are associated with the two characteristic velocities $\lambda_1 = u - c$ and $\lambda_2 = u + c$ respectively. One other equation is needed so that the three flow variables can be calculated. We choose $S = \ln(p/\rho^\gamma)$ where S is entropy. For subsonic inflow $u < c$ characteristic velocity $\lambda_2 > 0$ carrys information into the domain and therefore the characteristic variable R_2 can be specified along with one other condition. The Riemann invariant R_2, and S are set to exact values. The other characteristic velocity $\lambda_1 < 0$ carries information outside the domain and therefore, R_1 is extrapolated from the interior flow

variables. On subsonic outflow $u < c$ and $\lambda_2 > 0$ carries information outside the domain, while $\lambda_1 < 0$ propagates into the domain, so only R_1 is fixed to exact values and R_2, and $\ln(S)$ are extrapolated. Once these three variables are available at the boundary the three flow variables $\mathbf{Q}$ can be obtained. If we consider the boundary procedure as an operator on the interior data, we can cast the boundary scheme as

$$B(\mathbf{Q})_i = \mathbf{Q}_i - \mathbf{B}(\mathbf{Q}_{i+1}) = 0 \quad i = 0$$

and

$$B(\mathbf{Q})_i = \mathbf{Q}_i - \mathbf{B}(\mathbf{Q}_{i-1}) = 0 \quad i = J_N,$$

which are nonlinear equations at the boundaries.

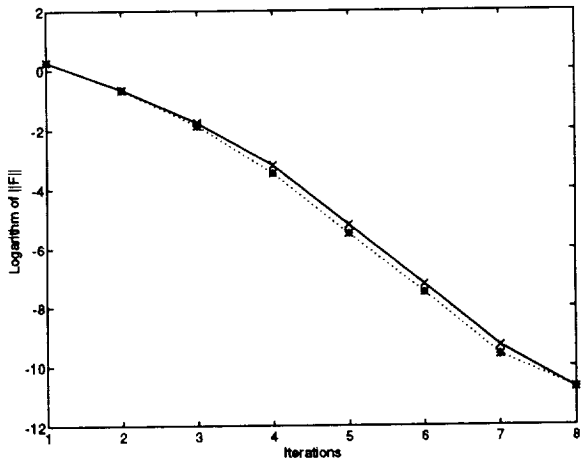

Figure 10: Results for 1D Euler using unlimited dissipation, Solid Line: tensor-GMRES; Dotted-line: Newton-GMRES.

The total system we shall solve is

$$\mathcal{F}(\mathbf{Q}) = \begin{cases} \delta_x \mathbf{E}(\mathbf{Q})_j - \mathbf{H}(\mathbf{Q})_j + D_j^2(\mathbf{Q}) + D_j^4(\mathbf{Q}), & j = 1, \cdots, J_N \\ B(\mathbf{Q})_i = 0, & i = 0, J_N \end{cases}$$

The Jacobian matrix for (5.13) is obtained in two ways. An approximated Jacobian is formed analytically except, where due to the non-differential form of the ϵ's, the nonlinear coefficients for the artificial dissipations, D^2 and D^4 are frozen at the linearized state, i.e., they are not linearized. In another form, the Jacobian is obtained through a Frèchet derivative, where error tolerances are appropriately chosen. The results presented below are basicly independent of the choice of Jacobian linearization. The order of the system is $N = (J_N + 1) \times 3$. A key element of the success of the solution using the Krylov subspace methods is the choice of preconditioning. This issue for systems such as (5.13), which are not diagonally dominate, is not straightforward and is still the subject of active research.

We shall not go into the details of the preconditioner here, and only state that the same preconditioner is used for both the Algorithm NG and TG so that consistent comparisons can be made.

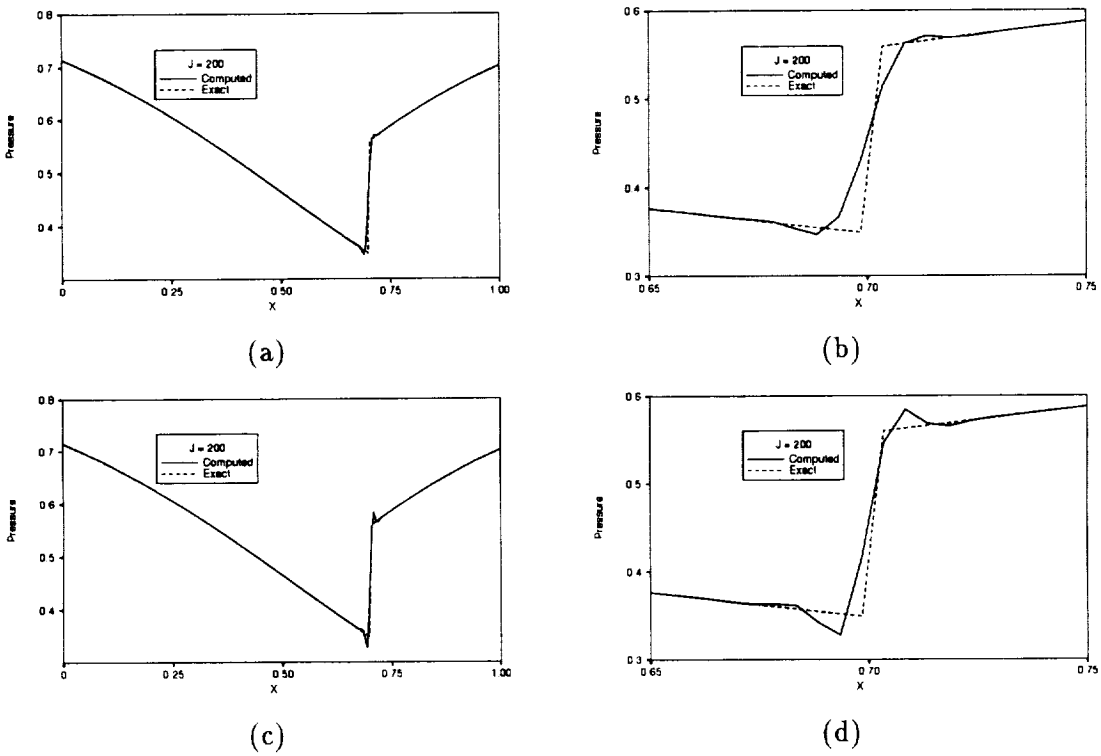

Figure 11: Pressure distribution: Solid Line: Computed; Dotted Line: Exact. (a) and (b) Full Nonlinear Dissipation; (c) and (d) Unlimited Dissipation.

Figure 9 shows Algorithm NG and Algorithm TG applied to (5.13) for $J_N = 200$; $N = 603$. In the case of Algorithm NG the convergence appears linear taking approximately 125 steps to converge, while Algorithm TG shows about a factor of 6 decrease in the number of nonlinear iterations and appears to be at least fast linear. To date, our analysis indicates that the system derived form (5.13) is nonsingular and so we do not consider this an example similar to the singular ones presented above. But, we have demonstrated, at least numerically at this time, that the nondifferential nature of the nonlinear dissipation coefficients in (5.11a–5.11d) is the source of the linear behavior observed. Figure 10 shows the convergence results with $\kappa_2 = 0$, resulting in a quadratic-like convergence from both Algorithm NG and TG. Nonlinear switching, such as defined in (5.11a–5.11d), is typical of current numerical algorithms for the Euler and Navier-Stokes equations. They may take a similar form to (5.11a–5.11d), see [22] or be in the form of limiters for upwind techniques, e.g. [28],[16]. The nonlinear switching (limiting) is necessary to eliminate overshoots at shocks, where higher order schemes are limited to lower order which more correctly differences the

equations across discontinuities. Figure 11, shows the results with and without limiting, notice the overshoots across the discontinuity which are more pronounced for the unlimited case. In general, some form of limiting will be required and in these cases Algorithm TG appears to be capable of at least fast linear convergence in contrast to Algorithm NG's slow linear convergence.

5.5 Summary and discussion of test results

The test results of this section indicate that the tensor-GMRES algorithm that we tested is more efficient in number of nonlinear iterations than the analogous Newton-GMRES method on singular and nonsingular problems, and significantly more efficient on problems where the Jacobian matrix has a small rank deficiency at the solution or the function has discontinuity. Since no failure was detected for either of the methods in the tests, it is inconclusive which method would be more robust. We observed that for each nonlinear iteration, the number of restarts of the GMRES algorithm for solving the Newton equation was ranging from 3 to 20 (recall that for all test problems the number of Arnoldi iterations allowed in the GMRES algorithm was set to 20). As discussed in Section 4, in the tensor algorithm the computation of the tensor step only uses the Krylov subspace that is produced by the last restarted GMRES algorithm. Statistically, the size of the last Krylov subspace that is used in the tensor model formation and solution is likely to be half of the number of Arnoldi iterations allowed in the GMRES algorithm (for example $20/2 = 10$ for our tests). As analyzed in Subsection 4.7, the extra computational cost of each tensor iteration is only a fraction of the cost of the last restarted GMRES algorithm. Hence for our test problems (some of them are easy to solve), statistically, the extra cost of one tensor-GMRES iteration would be ranging from 2.6% (when 20 restarts of GMRES were needed) to 20% (when 3 restarts of GMRES were needed), compared to the cost of one Newton-GMRES iteration. In addition, from our experience the tensor-GMRES did not generate iterates where the Newton equations would be harder to solve. Hence the savings of the tensor-GMRES algorithm in number of nonlinear iterations can be roughly translated into savings in overall computational costs, especially for problems where more restarts of the GMRES algorithm were needed to solve the Newton equations. In general, at each iteration, the more restarts of the GMRES algorithm that are required for solving the Newton equation, the lower the extra cost for a tensor-GMRES step. For real world problems, it is likely that at each nonlinear iteration, the solution of the Newton equation would require a significant number of restarts of the GMRES algorithm. Hence the extra cost of the tensor-GMRES method is likely to be minimal in practice.

6 Summary and topics for future research

This paper has introduced the tensor-GMRES method for systems of nonlinear equations. This method has similar requirement for storage and arithmetic per iteration to the Newton-GMRES method. This method is also consistent with preconditioning and matrix free

implementation. An implementation of full nonlinear algorithm using the tensor-GMRES method has shown to be more efficient on both nonsingular and singular problems than analogous implementation of the Newton-GMRES method. The efficiency advantage of the tensor-GMRES method is particularly large on problems where the Newton-GMRES method exhibits linear convergence (due to singularity or discontinuity).

Based on these results, it would appear worthwhile to continue research on tensor-Krylov methods on nonlinear equations. The two main topics for future research would appear to be practical implementation and farther testing of the tensor-GMRES methods for nonlinear equations, and new tensor-Krylov methods for nonlinear equations. We discuss each of these briefly.

As seen in Section 5, our implementation is still in early stage. Several directions can be pursued immediately to improve the current implementation: (1) Scaling in both the variable space and the function space can be added; (2) Matrix free implementation of the tensor-GMRES method, which can be achieved in a fashion similar to analogous implementation of the Newton-GMRES method, can be pursued; (3) More sophisticated stopping criteria in the nonlinear algorithm can be included; (4) More global convergence strategies such as model trust region techniques can be integrated. We would like to continue our testing of the tensor-GMRES method on more practical problems. One interesting task is to test the tensor-GMRES method on the ARC2D code [21] that is the two dimension version of the ARC1D code that we tested in Section 5.

Secondly, new tensor-Krylov methods can be developed. An immediate direction that one can pursue is a tensor-Arnoldi method since the Arnoldi's method for linear systems is closely related to the GMRES method for linear systems. A less straightforward direction that can be pursued in the future is to combine tensor methods with Krylov methods that use two mutually orthogonal sequences such as BiCG and QMR. We are currently investigating this possibility.

Acknowledgement

We thank Prof. Bobby Schnabel for many helpful discussions related to this paper.

References

[1] W. E. ARNOLDI, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[2] A. BOUARICHA, *Solving large sparse systems of nonlinear equations and nonlinear least squares problems using tensor methods on sequential and parallel computers*, PhD thesis, Department of Computer Science, University of Colorado, 1992.

[3] P. N. BROWN AND A. C. HINDMARSH, *Reduced storage methods in stiff ODE systems*, J. Appl. Math. Comput., 31 (1989), pp. 40–91.

[4] P. N. Brown and Y. Saad, *Global convergent techniques in nonlinear Newton-Krylov algorithms*, Tech. Rep. 89.57, Research Institute for Advanced Computer Science, NASA Ames Research Center, November 1989.

[5] ——, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 450–481.

[6] T. F. Chan and K. R. Jackson, *The use of iterative linear equation solvers in codes for large systems of stiff IVPs for ODEs*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 378–417.

[7] D. W. Decker, H. B. Keller, and C. T. Kelley, *Convergence rate for Newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 296–314.

[8] D. W. Decker and C. T. Kelley, *Newton's method at singular points I*, SIAM J. Numer. Anal., 17 (1980), pp. 66–70.

[9] ——, *Newton's method at singular points II*, SIAM J. Numer. Anal., 17 (1980), pp. 465–471.

[10] ——, *Convergence acceleration for Newton's method at singular points*, SIAM J. Numer. Anal., 19 (1981), pp. 219–229.

[11] J. E. Dennis Jr. and R. B. Schnabel, *Numerical methods for nonlinear equations and unconstrained optimization*, Prentice-Hall, Englewood Cliffs, NJ., 1983.

[12] D. Feng, *Tensor methods for constrained optimization*, PhD thesis, Department of Computer Science, University of Colorado, 1993.

[13] D. Feng, P. D. Frank, and R. B. Schnabel, *Local convergence analysis of tensor methods for nonlinear equations*, Mathematical Programming, 62 (1993), pp. 427–459.

[14] A. Griewank, *On solving nonlinear equations with simple singularities or nearly singular solutions*, SIAM Review, 27 (1985), pp. 537–563.

[15] A. Griewank and M. R. Osborne, *Analysis of Newton's method at irregular singularities*, SIAM J. Numer. Anal., 20 (1983), pp. 747–773.

[16] A. Harten, *High resolution schemes for hyperbolic conservation laws*, J. Comp. Phys., 49 (1983), pp. 357–393.

[17] C. T. Kelley and R. Suresh, *A new acceleration method for newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 1001–1009.

[18] J. J. Moré, *A collection of nolinear model problems*, Lectures in Applied Mathematics, 26 (1990), pp. 723–762.

[19] J. J. MORÉ, B. S. GARBOW, AND K. E. HILLSTROM, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software, 7 (1981), pp. 17–41.

[20] J. M. ORTEGA, *Numerical Analysis*, Academic Press, New York, 1972.

[21] T. H. PULLIAM, *Efficient solution methods for the navier-stokes equations.* Lecture Notes for the von Kármán Institute For Fluid Dynamics Lecture Series : Numerical Techniques for Viscous Flow Computation In Turbomachinery Bladings, von Kármán Institute, Rhode-St-Genese, Belgium , 1985.

[22] ——, *Artificial dissipation models for the euler equations*, AIAA J., 24 (1986), pp. 1931–1940.

[23] G. W. REDDIEN, *On Newton's method for singular problems*, SIAM J. Numer. Anal., 15 (1978), pp. 993–996.

[24] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[25] R. B. SCHNABEL AND A. BOUARICHA, *Tensor methods for solving large sparse systems of nonlinear equations and nonlinear least square problems.* In preparation.

[26] R. B. SCHNABEL AND P. D. FRANK, *Tensor methods for nonlinear equations*, SIAM J. Numer. Anal., 21 (1984), pp. 815–843.

[27] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Review, 15 (1973), pp. 727–764.

[28] P. K. SWEBY, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Num. Anaylsis, 21 (1984), pp. 995–1011.